

SIMULATION OF TDC-12 ON IBM-7044

A thesis submitted

In Partial Fulfilment of the requirements

for the Degree of

Master of TECHNOLOGY

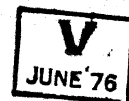
by

RAJJAN SHINGHAL

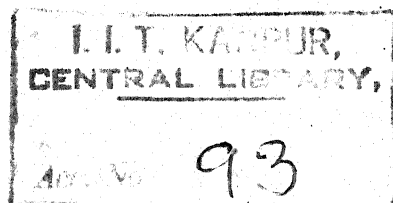
TO THE

Department of Electrical Engineering

Indian Institute of Technology, Kanpur



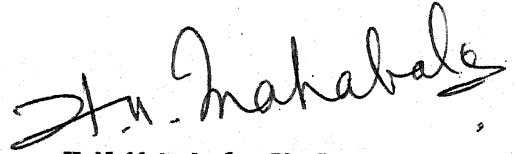
JUNE 1968.



Thesis
510.7834
Sh 63 /s

EE-1968-M-SHI-SIM

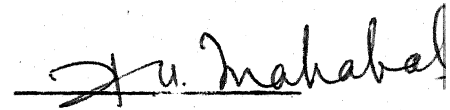
This is to certify that this simulation of TDC-12 on IBM-7044 has been carried out under my supervision and has not been submitted elsewhere for a degree.



H.N. Mahabala, Ph.D.
Asst. Professor,
Department of Electrical Engineering,
Indian Institute of Technology, Kanpur.

This thesis has been approved for the award of the Degree of Master of Technology in accordance with the regulations of the Indian Institute of Technology, Kanpur.

H.N. Mahabala, Ph.D.,
Asst. Professor,
Department of Electrical Engineering,
IIT-Kanpur.



ACKNOWLEDGEMENTS

I wish to thank Dr.H.N.Mahabala for his guidance and helpful criticism throughout the course of this endeavour.

Thanks are also due to Dr.V.Rajaraman for his keen interest and suggestions.

The assistance given by the Staff,Computer Centre , IIT-Kanpur, is gratefully acknowledged.

RAJJAN SHINGHAL

LIST OF CONTENTS

	Page
Synopsis	1
1. Introduction	2
2. The TDC-12	4
2.1 Machine Features	4
2.2 The Central Processor	6
2.2.1 Accumulator	6
2.2.2 Carry Register	6
2.2.3 Program Address Counter	6
2.2.4 Memory Address Register	7
2.2.5 Switch Register	7
2.2.6 Memory Data Register	7
2.2.7 Instruction Register	7
2.2.8 Auto-Indexing	7
2.3 Input/Output	7
2.4 Program Interrupt	8
2.5 Data Interrupt	9
3. Simulation Techniques	10
4. The TDC-12 Simulator	13
4.1 The Central Processor	13
4.2 Input/Output	14
4.2.1 Use of Input Devices	14
4.2.2 Use of Output Devices	15

	Page
4.3 Program Interrupt	16
4.4 Additional Features	17
4.4.1 Input-BCD or Binary	17
4.4.2 Output Memory Dump	18
4.4.3 Trace Feature	19
4.4.4 Clock	19
5. Conclusions	20
List of References	21
<u>Appendices</u>	22
1. (A) Storage Reference Instructions	23
(B) Non-Storage Reference Instructions	26
(C) Register Set-1	27
(D) Register Set-2	30
(E) Input Output Instructions	32
2. The Simulator Subroutines	37
3. The Major State Generator	39
4. Using the Simulator	41

LIST OF FIGURES

	Page
1. Format Data Word Storage	5
2. Format Instruction Word Storage	5
3. Format Storage Reference Instructions	24
4. Format RS-1 Instruction Set	27
5. Format RS-2 Instruction Set	30
6. Input/Output Instruction Set	32
7. Logical Design of TDC-12	42
8. Flow Chart for the simulator of TDC-12 on IBM-7044	43

LIST OF TABLES

	Page
1. Octal Codes for I/O Units of IBM-7044 and TDC-12	47
2. Sample Result and Listing of Simulator	49

LIST OF ABBREVIATIONS

ACCM	..	ACCUMULATOR
BCD	..	BINARY CODED DECIMAL
Cols	..	COLUMNS
CRYRG	..	CARRY REGISTER
DDL	..	DEVICE DERAILMENT LOCATION
MAP	..	MACRO-ASSEMBLY PROGRAM
MAR	..	MEMORY ADDRESS REGISTER
MDR	..	MEMORY DATA REGISTER
MSG	..	MAJOR STATE GENERATOR
PAC	..	PROGRAM ADDRESS COUNTER
RM	..	RECORD MARK
RS-1	..	REGISTER SET-1
RS-2	..	REGISTER SET-2
SWRG	..	SWITCH REGISTER
I/O	..	INPUT/OUTPUT

SYNOPSIS

1

SIMULATION OF TDC-12 ON IBM-7044

A thesis submitted in partial fulfilment of the requirements for the Degree of Master of Technology

by

RAJJAN SHINGHAL

to the Department of Electrical Engineering,
Indian Institute of Technology, Kanpur.

JUNE 1968.

A simulator for a small on-line stored-program computer (similar to TDC-12 being developed by the Bhabha Atomic Energy Commission) on IBM-7044 has been written in the MAP language. It is mainly an aid to write the assembler, compiler and Utility Library Programs for the on-line computer.

This report discusses in the beginning, why a simulator is necessary. The features of the on-line computer have been given. The simulator was written at the Hardware level. The simulator includes in it the Program Interrupt feature, which enables the computer to attend I/O interrupt on a priority basis. For I/O two teletypes are simulated. To ensure real time compatibility between the actual computer and its simulator, a Pseudo-Clock which keeps record of time taken for execution of program in number of machine cycles, has been included.

A Pseudo-Load Routine ensures fast storing of program. Debugging aids in form of trace feature and the facility of dumping out memory of the simulated computer are also included.

A computer system cannot be thought of as only the hardware unit; the software is an integral part of it. To facilitate the use of higher level languages, it is essential to develop a powerful software for the computer system. The development of software is time-consuming and expensive. For most of the modern systems about sixty percent of the development cost goes to develop software. The first step in writing the software is to write an assembler. The minimal software provides a take-off stage for the improvement and development of further software.

An early method to write the assembler for a proposed computer system employs a bootstrap technique. After the computer system is ready, a minimum assembler is written in machine language. Using this minimum assembler language another assembler is written to accept higher-level-assembly-language programs. The second assembler is then compiled into machine language by the first assembler, thus producing an Updated Assembler-II which can accept Assembly Language-II programs. Successive utilization of this technique provides a maximum assembler. This method requires considerable amount of machine-language programming, which is very tedious and time consuming.

It is desirable to have the software of the proposed computer system operational, by the time the hardware circuitry is ready. For this an existing computer can be used. As a computer system is developed it is essential to have means for evaluating different hardware configurations from point of view of efficiency of software. This can be done by digital computer simulation of the

proposed computer system. The constraints under which the simulated system will operate have to be clearly defined.

In simulation of a system a model is to be realised whose behaviour in the specified environment is the same as that of the original system. Then the response of the model to a specified stimulus condition is utilized in deducing corresponding conclusions about the system under simulation. Once the simulator for the proposed computer system is ready, the Assembler, Compiler and Utility Library Programs for it can be written. Moreover while working on the simulator one may come across areas, specially in choice of machine operation codes, in which improvements may be suggested to the hardware designer.

Thus by the time the hardware circuitry of the proposed computer is ready, its completely debugged software is also ready. This saves time and also ensures ease in developing the software. Simulator studies are an essential part of developing a computer system and its software.

The Bhabha Atomic Energy Commission is developing a small on-line, stored-program computer----- TDC-12. It is a general purpose computer meant for,

1. System and Control Application (i.e. Real Time Application)
2. On-Line data collection and Reduction
3. Limited Computation

2.1 MACHINE FEATURES

Word Format	Binary ✓
Word Length	12 bits
Memory Capacity	4096 words
Arithmetic Used	Twos complementary
Mode of Operation	Parallel Synchronous
Time for one machine cycle	1.5 microseconds

Memory Organization

Since the system has a 4096 word core-memory 12 bits are required to address all the locations. To reduce the number of bits required for addressing, the memory is divided into sectors of 64 words each. The sectors are numbered 0 through 63 and the locations in each sector are also numbered 0 through 63.

The hardware does not provide for multiplication, Division and Floating-Point Computation. Only Integer Arithmetic can be done. Anyhow by writing suitable software routines multiplication, division and Floating-point computation can be carried out. Similarly software routines could be written to perform the operations of square root, sine, cosine, Arctangent, natural logarithm and exponential.

Information in the memory core may be stored either as Data word or as an Instruction. The data word in twos complementary form is stored as in Figure-1.

The instruction set consists of 12 Storage Reference and 3 Non-Storage Reference instructions. Storage-Reference instructions

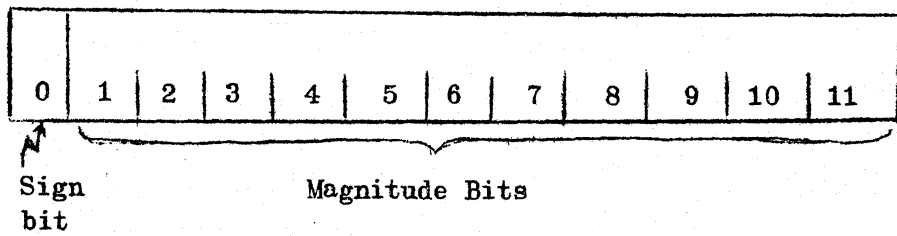


Figure-1 Format- Data Word Storage

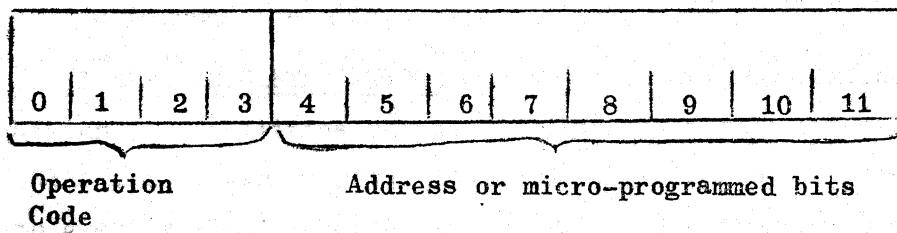


Figure-2 Format- Instruction Word Storage

store or retrieve data from the core memory while others do not. Bits 0 through 3 specify operation code in all instructions. The non-storage reference instructions could be microprogrammed to perform several operations in one instruction. For more detailed explanation see Appendix I. The format for an instruction word storage in core memory is shown in Figure-2.

2.2 THE CENTRAL PROCESSOR The logical arithmetic, data processing and control functions and storing and retrieving information for core memory are performed in the central processor.

2.2.1 ACCUMULATOR (ACCM) It is a 12 bit register where all arithmetic and logical operations are performed. It can be cleared or complemented, its contents can be circulated right or left with Carry Register under program control. The contents of MDR can be added to ACCM and results left in the ACCM. The contents of both these registers can be combined by logical operations AND or EXCLUSIVELY OR, the result being left in the ACCM. The INCLUSIVELY OR can be performed between ACCM and SWRG. The result remains in the ACCM. In I/O transfers information is transferred between core memory and the peripheral device through the ACCM.

2.2.2 CARRY REGISTER (CRYRG) This one bit register extends the arithmetic facilities of the ACCM. In two's complement arithmetic the CRYRG acts as an overflow indicator which can be checked by the program, to greatly simplify and speed up multi-precision arithmetic routines. The CRYRG can be cleared, complemented and its state sensed independent of the accumulator. It is included with the ACCM in circulate operations.

2.2.3 PROGRAM ADDRESS COUNTER (PAC) It is a 12 bit register and contains the address of the memory location which contains the next instruction to be executed. Information enters into the PAC from the core memory via the MDR, MAR or SWRG. Information in the PAC is transferred to MAR to determine the core memory address from which the next instruction is to be taken. Incrementation of PAC provides skipping of one instruction or two based upon a programmed test of information or conditions.

2.2.4 MEMORY ADDRESS REGISTER(MAR) It is a 12 bit register containing address of the memory location currently selected for reading or writing. All 4096 words of core-memory are directly addressable by this register. Data from PAC can be set into it.

2.2.5 SWITCH REGISTER(SWRG) Information can be placed in this 12 bit register by switches on the console of TDC-12.

2.2.6 MEMORY DATA REGISTER(MDR) It is a 12 bit register. All information transferred into or out of the core-memory passes through MDR. Information is read from a memory cell into MDR and rewritten in the cell. The contents of MDR can be incremented by 1.

2.2.7 INSTRUCTION REGISTER(IR) This 4 bit register contains the operation code of the instruction currently being executed by the computer. The four most significant bits of the current instruction are loaded into the IR from MDR during Instruction Fetch Cycle. The operation code bits are then decoded to produce fifteen basic instructions and affect the cycles and states entered at each step in the program.

2.2.8 Auto Indexing When a location between 10(octal) and 17(octal) in sector zero of core memory is addressed indirectly the contents of the location are read, incremented by one, rewritten in the same location and then taken as the effective address of the instruction. This provides the facility normally given in other computers by index registers. If location 14(octal) contains 2132 and if this is indirectly addressed the number 2133 is stored in location 14(octal) and the effective address is taken as 2133.

2.3 INPUT/OUTPUT

The Teletype is the unit for both input and output of information. On the user side, the information appears as type print and on the computer side it appears as 8 bit binary number. Table 1 in Appendix-5 gives the characters and their equivalent octal codes eg. letter C when typed on keyboard goes into ACCM as 303 (octal), after suitable instruction to read keyboard is given. Octal combinations for Line Feed and Carriage Return are also listed in Table-1. The teletype is slow as compared to the computer execution

speed. The computer must temporarily suspend execution of the current program or execute another program while the Teletype (rate 10 characters per second) is again in synchronism with it. When the Teletype is ready it sets a flag which is watched for by the computer. After the execution of the current instruction the next character is transferred and the flag is reset. An instruction is required for the transfer of each character.

2.4 PROGRAM INTERRUPT By this, the program control can be suspended from the current program and transferred to another routine of higher priority. The TDC-12 provides the interrupt feature with multi-level priority. When an interrupt occurs the contents of the PAC are stored in location 0000 and control is transferred to location 0001. A software subroutine stored from there should then sort the source of interrupt and transfer control to the proper subroutine. Each service subroutine enables only those interrupts which have higher priority. This solves the problem of interconnecting slow I/O devices to the fast computer. Facility is provided where one or more peripheral ^{Units} may be disabled from interrupting.

A computer may transmit a word to the printer which may require many milliseconds to be output. Rather than waiting and wasting time the computer can move to another program, and return to the printing program when the printer sends signals that it is ready to accept another character. This improves the efficiency of I/O operation.

Any interrupt can be enabled only if the Interrupt feature has been turned ON. After interrupt takes place it is turned OFF, and for future interrupts of higher order to occur, it should be turned

ON again, by the software subroutine stored from location 0001.

2.5 DATA INTERRUPT

This facility provides fast data transfers from and to core memory directly, with fast I/O devices like magnetic tapes. This has all the facilities of a Data Channel except that at the time of I/O computations cannot go on. The Interrupt is indicated by a request from the peripheral device (not by programmed instruction) and are interlaced with the program in progress. Thus the device may transfer a word with memory, whenever it is ready, without waiting for an instruction in the program.

The break or interrupt may be of two types:

1. registers in the device specify the core memory address of each transfer and count the number of transfers to determine the end of data blocks.
2. two computer memory locations perform these functions, simplifying the device interface by omitting hardware registers.

A simulator of the proposed computer system could be written at the Instruction Level. In this technique each instruction of the proposed computer is taken and decoded to achieve the end objective. This technique requires the simulation of the entire core memory but does not require the simulation of all the electronic registers. Only those electronic registers to which the user has direct access eg. the accumulator, need be simulated. The information flow does not follow the path and pattern of the proposed computer. It is like building the model of a system where only the end-product is the same but the internal operation may be different. The simulator is independent of the hardware logical design. Instruction level simulator may be efficient in terms of machine time but it may not be useful in examining modifications to hardware.

A better technique is the Hardware Level simulation. Here the entire core-memory and all the allied electronic registers are simulated. The instruction is decoded to achieve the end-objective, but at the same time the, the information flow follows very nearly the same path and pattern as in the proposed computer. Here not only the end-product is the same but all the interconnecting parts together with their mutual relationship are essentially the same as in the actual computer. This technique can be used to evaluate proposed changes in the system without actually incorporating the changes in the hardware. If a small change in the hardware is contemplated, a corresponding change in the simulator is first done and evaluated before fabrication. The utilization of the

various functional units can be studied. This technique thus provides reliability and flexibility. The simulator of TDC-12 on IBM-7044 was written on the Hardware Level.

An on-line computer is controlled by its environment and it is possible to interrupt the normal working of the computer and communicate with it through a peripheral device, at any required time. As compared to the speed of the Central Processing Unit, the peripheral devices are very slow, and while I/O occurs the computer may have to wait or execute another program. It becomes necessary for the simulator to follow the same pattern. The Hardware-Level simulator makes this possible as there is one-to-one correspondence between the simulator and the hardware of the system. This provides for a well organized and clearly defined simulator on which improvements are easy to implement.

For widest applications, a computer system simulator should be a dynamic simulator, that is it should have a clock whose advances correspond to real time advances in the simulated computer. The simulator should be able to output snap-shots of the state of the computer system. A memory-dump routine in the simulator provides the status of the core of the simulated computer, and the electronic registers. A ^aTrace feature is incorporated in the simulator to determine the actual sequence of instructions executed (only jumps need be indicated). A Pseudo-Clock is provided which is incremented suitably at the end of each machine cycle. This provides the time in object machine cycles which a program will take for execution. Since the simulated computer is meant for real time

application it is imperative to take time factors into account. Especially the Clock feature is indispensable. Whenever any of the I/O devices are operating there is considerable delay between successive I/O of characters. The Clock is then used to time the simulator.

With the languages available at the Computer Centre at Indian Institute of Technology, Kanpur the simulator could have been written in Fortran-IV, Algol or MAP. In MAP it is easy to handle individual characters and also individual bits of a word. Hence the simulator was written in MAP language.

4. THE TDC-12 SIMULATOR

4.1 THE CENTRAL PROCESSOR Locations 20000_8 through 27777_8 in IBM-7044 have been reserved as the 4096 word core-memory of TDC-12. These locations were chosen as the last four digits of these locations represent the address of the location as in TDC-12. Thus this ensures an easy correspondence in addresses of IBM-7044 and TDC-12. The 12 bit information pertaining to TDC-12 is always stored and interacted in the lower 12 bits of these locations. The electronic registers----- ACCM, CRYRG, PAC, MAR, SWRG, MDR and IR ---- are simulated by reserving one memory location for each of these. Their working conforms to that described earlier in the description of TDC-12.

For execution of any instruction the contents of PAC are placed in MAR. The contents of location in MAR are placed in MDR, and contents of PAC are incremented by 1. The instruction in MDR is decoded by a Decode routine. If the contents of MDR form an illegal instruction, a suitable message is printed. Further the contents of TDC-12 memory and allied electronic registers are dumped out and program is terminated. If it is a legal instruction it is decoded further, to find out whether it is a Storage Reference type or not. If it is a Non-Storage Reference type of instruction, microprogramming is tested for and suitable subroutines are called for execution. In the Storage Reference type of instruction, the effective address of the operand is determined after checking for sector bit, indirect addressing and auto-indexing. This may need an extra Read cycle. Suitable subroutines are then called for execution.

For each instruction that is executed the contents of the

CLOCK ARE INCREMENTED by the number of machine cycles the instruction takes for execution. If the TRACE feature has been turned ON, then whenever there is a skip or jump in program-execution-control as distinct from the sequential execution of a program, a message is given indicating the jump. At the successful completion of a program the contents of core-memory, electronic registers and the CLOCK are dumped out if Memory Dump Feature has been turned ON earlier.

If the Interrupt is turned ON then at the end of each instruction test is made if any of the peripheral devices are interrupting. If so, the contents of the PAC are stored in location 0000 and program control is transferred to location 0001. If none of the peripheral devices are interrupting, the execution of the same program goes on.

4.2 INPUT/OUTPUT Two teletypes are simulated for I/O. The ^ebreak-up of the devices is into four units which have been sequentially numbered.

<u>UNIT</u>	<u>NUMBER</u>	
Keyboard Reader	01	} Input Units
Paper Tape Reader	02	
Teleprinter	03	} Output Units
Paper Tape Punch	04	

A nominal modification in the simulator will ensure the addition of more I/O devices.

4.2.1 USE OF INPUT DEVICES For input the teletype handles character by character at the rate of 10 characters per second. The input is serial as an input instruction is to be executed for the transfer of every character from the input device to the ACCM. On IBM-7044 input is parallel as all the characters on a card can be read simultaneously. So that the input on IBM-7044 card reader corresponds to that of the teletype on TDC-12, the contents of the card are placed

in an input buffer. On an input instruction, the contents of the buffer are shifted to ACCM character by character. The rate of transfer of characters from buffer to ACCM in IBM-7044 is controlled by a program which uses the Clock, and it corresponds to that of the Teletype in TDC-12. When the buffer is exhausted and still more data is required a suitable message appears on the IBM typewriter. The data is read in parallel from a card which is then supplied internally serially. On every card the end of data is shown by a Record Mark (punch in rows 0-2-8 using the multiple punch). Data can utmost extend upto 72 columns with RM in column 73. Columns 77 and 78 should have 01 or 02 punched to signify which unit is supposed to be used.

4.2.2 USE OF OUTPUT DEVICES For output, the teletype handles character by character at the rate of 10 characters per second. Output is thus serial as an output instruction is required for each character. IBM-1403 the on-line printer connected to IBM-7044 is a parallel printer as an entire line upto 132 characters can be printed simultaneously. The teletype features had to be simulated on the on-line printer. On execution of an output instruction, one character is transferred at a time from ACCM to a buffer. The buffer can utmost hold 72 characters. When the buffer is full, the entire contents of the buffer are printed out automatically. If fewer than 72 characters are required to be printed in one line, instructions for Line Feed and Carriage Return are to be given. See Table-1 in Appendix-5 for their octal codes. After any printing the buffer is automatically cleared and is prepared to receive more information. The rate of

transfer of characters from ACCM to buffer corresponds to the output rate in teletype in TDC-12. After printing a line an indication is given signifying which unit is supposed to output. A typical output from Unit 03 will look like:

1234567890QWERTYUIOPASDFGHJKLZXCVBNM--- --

UNIT 03

4.3 PROGRAM INTERRUPT The execution of a program can be suspended at any point and control can be transferred to another routine of higher priority by the interrupt feature. After execution of the service routine, control may be returned to the original program. Any one or more of the I/O devices may cause interrupt. Each one ^{of} the devices has a memory location called the Device Derailment Location (DDL) which is used to indicate the interrupt by that device. The allotment of the DDLs is as follows,

UNIT	TDC-12 Location in Octal Notation	IBM-7044 Location in Octal Notation
01	7777	27777
02	7776	27776
03	7775	27775
04	7774	27774

DEVICE

Presence of a 1 in the lowest bit of a DDL indicates that the corresponding device is interrupting. Thus if during the execution of a program an interrupt is to be caused put IBM-7044 in manual mode. Enter through keys a 1 in the lowest bit of the corresponding DDL. Put into automatic mode and press START. If the Interrupt had been turned ON earlier, then after the execution of the current instruction interrupt shall occur. The contents of PAC are stored in location 0000 and control is transferred to location 0001. Presence of a 1 in bit 5 of any DDL (viewed as in TDC-12) disables the corresponding device from interrupting. Thus the routine for

sorting out source of interrupt which is stored from location 0001 must check for this.

4.4 ADDITIONAL FEATURES As mentioned earlier the I/O devices connected to TDC-12 are slow. If the program to be run is fed by these devices, then it takes a long time to be stored in the TDC-12 memory. Also to output a character will be very slow. Hence as an additional feature facility exists for fast input and output.

4.4.1 INPUT-----BCD OR BINARY A Pseudo-Load routine takes care of this. Input is punched on cards. The first card is a HEADER card and the program to be stored is punched on subsequent cards either in Binary Coded Decimal (BCD) or Binary form.

BCD MODE On Header card punch BCDM in Columns 3 through 6. The program to be stored is punched as follows,

Columns 1,2	Cols. 3 through 6	Cols. 7,8	Cols. 9 through 12	Cols. 14 onwards
Blank	AAAA	Blank	BBBB	Remarks

AAAA specifies where the information BBBB is to be stored. Both are in Octal Form. On the last card at the end of the program, field AAAA has DEND punched in it and field BBBB has the starting location counter where program control will be transferred for execution.

BINARY MODE On the Header card punch BNRV in Columns 3 through 6. For the main program each card has a 7-9 punch in Column 1. Column 2 has the number of TDC-12 instructions punched on that card. Column 3 has the location where the first instruction (appearing from Cols. 4) will be loaded. The rest of the instructions on the card are loaded in sequential order after that. Each instruction occupies one

column on the input card. All input is in octal form. The entire card upto 80 columns can be used. At the end of a program the last card is indicated by a 12-7-9 punch in Column 1. The address of the location where program control is to be transferred for execution is in Column 2. Rest of the card may be blank as it is ignored.

If on the Header card Columns 3 through 6 have neither BCDM nor BNEY punched in, then a message appears:

ILLEGAL INPUT, UNABLE TO READ

After dumping out the entire memory of TDC-12, ^{the} job is terminated.

4.4.2 OUTPUT---MEMORY DUMP This provides the Dump of the entire core-memory of TDC-12 together with all the simulated electronic registers including the CLOCK. This is a powerful debugging aid.

In the memory dump all output is in octal. The contents of PAC, MAR, CRYRG, SWRG and CLOCK are dumped without the necessary sign bit. For ACCM and MDR the sign bit is added. Contents of sixteen (20 octal) memory locations are printed in one line with the address of the first location in the line being printed out extreme left. If it is a valid operation code, the equivalent mnemonic is printed underneath the numerical contents. In the Storage Reference instructions the presence of a star(*) after the mnemonic represents indirect addressing. Similarly the presence of a 1 means the sector bit was 1, the absence of it means the sector bit was zero. If the contents of a location do not tally to a valid operation code then the contents are printed out with the proper sign. It may happen that consecutive blocks of sixteen locations have the same contents. In this case rather than print out the same contents over and over again, a message is given:

LOCATION XXXX THRU YYYY ALL CONTAIN ZZZZ

QQQQ

In case ZZZZ forms a legal instruction, QQQQ is the equivalent operation code, otherwise it is left blank.

Whenever during the execution of a program an illegal instruction is encountered, a message is given out:

ILLEGAL TDC INSTRUCTION. JOB TERMINATED.

The memory of TDC-12 is dumped out and execution halts.

It may happen that a program is completed successfully yet further improvements are to be made. For this the status of the core-memory and the electronic registers may be required. In that case on the Header card punch MMRY in Columns 15 through 18. Before terminating, the memory is dumped out. In case this facility is not required, the corresponding columns on the Header card may be left blank.

4.4.3 TRACE FEATURE This is also a debugging aid. On the Header card punch TRAS in columns 9 through 12. Whenever there is an abrupt change in the instruction location counter--as distinct from the normal sequential progress of the program---a message is printed out,
 LOCATION COUNTER JUMP XXXX THRU YYYY
 meaning thereby, that program control changed from location XXXX to YYYY. Thus this message is given on a JUMP, SKIP or on the successful outcome of a conditional skip. If this facility is not required leave the corresponding columns on the Header card blank.

4.4.4 CLOCK In IBM-7044 a location is reserved which increments during the execution of an instruction by as many cycles as the instruction takes on TDC-12. This helps in:

1. keeping track of time taken on TDC-12
2. Synchronizing the slow I/O devices with the main program execution.

This location is referred to as a clock.

5. CONCLUSIONS

In general, computer simulation provides a means for studying Systems. It can be applied to a wide variety of systems, both real and hypothetical. It is a very powerful research technique. The simulation provides a tool which could quickly and efficiently assist in the investigation and study of the performance of the proposed system.

The simulation of the small on-line computer consisted in writing a simulator in MAP language for IBM-7044. The simulator will be used in writing the software for the proposed computer. It is expected that the software may dictate eliminating some machine operation codes in favour of some others. The changes can be incorporated very easily in the simulator. Since the organization of the memory of the on-line computer is of the sector-type valuable experience in writing programs for such a machine (even before writing the software) can be obtained by the use of the simulator. The Clock and the trace feature will assist in optimizing programs from point of view of time and memory required.

LIST OF REFERENCES

1. S.N.Verma, System Design of Trombay Digital Computer TDC-12, Govt. of India, Atomic Energy Commission.
2. T.N.Reddy, Assembly System, Govt. of India, Atomic Energy Commission.
3. Digital Equipment Corporation, Small Computer Handbook, New PDP-8/i
4. J.Martin, Programming Real Time Computer Systems, Prentice Hall Inc.
5. Ivan Flores, The Logic of Computer Arithmetic, Prentice Hall Inc.
6. Ivan Flores, Computer Software, Prentice Hall Inc.
7. L.Rowell Huesmann and Rober P.Goldberg, Evaluating Computer Systems through Simulation, The Computer Journal, August 1967, Page 150.
8. N.R.Nielson, Simulation of Time Sharing Systems, Communications of A.C.M., Volume 10, Number 7, July 1967. Page 397.
9. G.W.Evans, Graham F.Wallace, G.Wallace, G.L.Sutherland, Simulation Using Digital Computers, Prentice Hall Inc.

A P P E N D I C E S

INSTRUCTION WORDS

The instruction(command)word specifies the instruction to be executed. Instruction words are of the following types:

- (a)..Storage Reference Instructions
- (b)..Non-Storage Reference Instructions
 - (i) Input Output Instructions
 - (ii) Register Instructions

Storage reference Instructions store or retrieve data from the core memory, while others do not. Bits 0 through 3 are used to specify operation code in all instructions.

(A) STORAGE REFERENCE INSTRUCTIONS

There are 12 of these instructions. Each instruction consists of two parts:

- (a)..bits 0 through 3 forming operation code.
- (b)..bits 6 through 11 forming address field.

The word format for these instructions is shown in Figure 3.

A 1 in bit 4 means indirect addressing, thereby specifying that the correct operand address is to be obtained from the location whose address is given in the address field. If bit 5 in the instruction word contains a 1, the six address bits (6 through 11) can address any location in the sector in which the current instruction is located. If bit 5 contains a zero, any location in sector zero can be addressed directly from any sector of core-memory. All other locations can be addressed indirectly by placing a 1 in bit 4 and placing six bit address in the instruction to specify the location in current

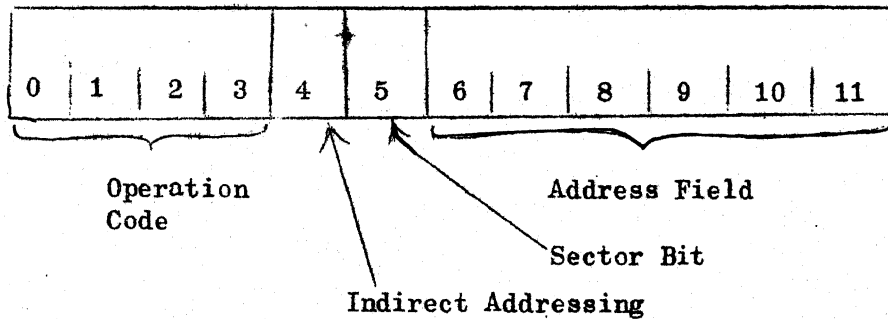


Figure-3 Format Storage Reference Instructions

sector or sector zero which contains the 12 bit effective address of the operand.

A list of Storage Reference Instructions is given. For all instructions except the JMP instruction time taken is 2 machine cycles for directly addressed and 3 machine cycles for indirectly addressed instructions. In JMP instruction the corresponding time is 1 and 2 machine cycles.

LOGICAL AND

Octal Code 04

Mnemonic AND Y

The logical operation AND is performed between the contents of location Y and that of ACCM. The result is left in the ACCM, the original contents of the ACCM being lost. The contents of memory location Y are unchanged.

Exclusive Or

Octal Code 10

Mnemonic XOR Y

The logical operation Exclusive OR is performed between the contents of memory location Y and that of ACCM. The result is left

in the ACCM and original contents of ACCM are lost. Contents of memory location Y are unchanged. Corresponding bits while ORing are compared independently.

LOAD ACCUMULATOR

Octal Code 14

Mnemonic LAC Y

The contents of memory location Y are loaded into the accumulator. The previous contents of the accumulator are lost. The contents of memory location Y are unchanged.

STORE ACCUMULATOR

OCTAL CODE 20

Mnemonic SAC Y

The contents of ACCM are stored in location Y. The previous contents of Y are lost. The contents of the ACCM are unchanged.

ADD

Octal Code 24

Mnemonic ADD Y

The contents of memory location Y are added to the ACCM in twos complement arithmetic. The result is left in the ACCM and the original contents of the ACCM are lost. The contents of location Y are unchanged. The CRYRG is set to 1 to indicate an overflow.

SUBTRACT

Octal Code 30

Mnemonic SUB Y

The contents of memory location Y are subtracted from the contents of the ACCM in twos complement arithmetic. The result is left in the ACCM and the original contents of the ACCM are lost. Contents of location Y are unchanged. The CRYRG is set to 1 to indicate an overflow.

REPLACE ADD MEMORY

Octal Code 34

Mnemonic RAD Y

The contents of location Y are added to the ACCM in twos complement arithmetic. The result is placed back in location Y, whose original contents are lost. The ACCM will also contain the sum and its original contents are lost too.

INCREMENT AND SKIP IF ZERO

Octal Code 40

Mnemonic ISZ Y

The contents of location Y are incremented by 1 in twos complement arithmetic. If the resultant contents of Y equal zero, the next sequential instruction is skipped. If the resultant contents of location Y are not equal to zero, the program proceeds to the next sequential instruction.

JUMP

Octal Code 44

Mnemonic JMP Y

The address Y is set into PAC so that the next instruction is taken from core memory location Y. Original contents of PAC are lost. Contents of ACCM are unaffected.

JUMP TO SUBROUTINE

Octal Code 50

Mnemonic JMS Y

The contents of the PAC are deposited in memory location Y and the next instruction is taken from location Y+1. Contents of ACCM are unaffected.

Compare Accumulator and Skip

Octal Code 54

Mnemonic CAS Y

The contents of ACCM are algebraically compared with the content of location Y. If the contents of ACCM ^{are} greater than contents of location Y, the next sequential instruction is executed. If the contents of ACCM are equal to the contents of location Y, the next sequential instruction is skipped. If the contents of the ACCM are less than contents of location Y, the next two sequential instructions are skipped. Contents of ACCM and contents of location Y are unchanged.

EXECUTE

Octal Code 60

Mnemonic XCT Y

The instruction in memory location Y is executed without changing program control (unless the instruction in location Y is a ^{is} JMP or JMS instruction). When the instruction in location Y ^{is} the next address stored is the address of the Execute instruction plus one. Effectively this is a one instruction subroutine.

(B) NON-STORAGE REFERENCE INSTRUCTIONS

There are two types of instructions which do not refer to any memory location. These are the Input/Output instructions and the Register Set instructions. Bits 0 through 3 represent operation code. Bits 4 through 11 serve as an extension of operation code and could be micro-programmed to perform several operations within one instruction. Each instruction is discussed separately. All these instructions are executed in one machine cycle.

There are two Register-Instruction sets, RS-1 and RS-2. RS-1 (operation Codes 70) is principally for clear, complement, rotate and increment operations. RS-2 (Operation Codes 74) is used principally in checking the contents of the ACCM and Carry Register and continuing to, or skipping the next instruction based on the check.

(C) REGISTER SET -1

The micro-instruction format is shown in Figure-4. Subsequently all the instructions are explained.

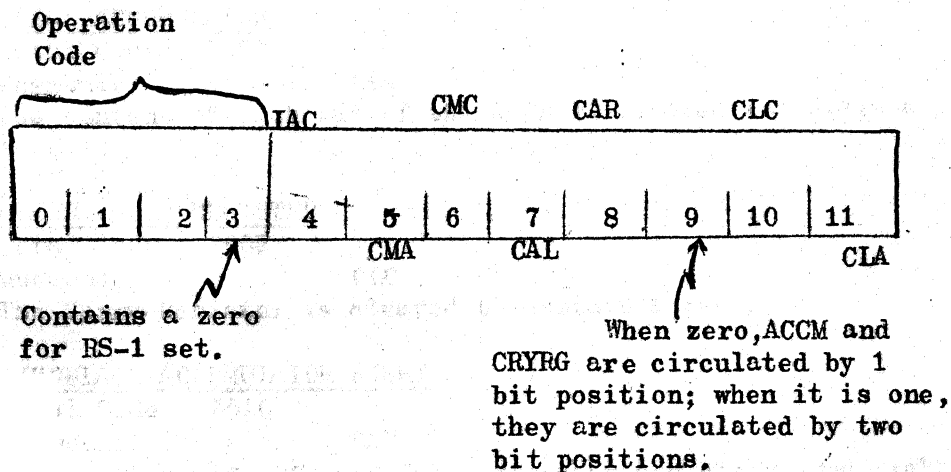


Figure-4 Format. RS-1 Instruction Set

Any logical combination of bits within the group can be combined into one micro-instruction eg. bits 4,5 and 10 can be combined but bits 7 and 8 cannot be combined as they represent conflicting operations. When micro-programmed the logical sequence in which instructions are executed are:

1. CLA, CLC
2. CMA, CMC
3. IAC
4. CAR, CAL, CTR, CTL

NO OPERATION

Octal Code 7000

Mnemonic NOP

This causes a one cycle delay in the program and the next instruction is initiated. It is used to add execution time to a program, such as to synchronize subroutine or loop timing with peripheral equipment timing.

CLEAR ACCUMULATOR

Octal Code 7001

Mnemonic CLA

The content of each bit of the ACCM is cleared to obtain a binary zero.

CLEAR CARRY REGISTER

Octal Code 7002

Mnemonic CLC

The Carry Register is cleared to contain a zero.

CIRCULATE ACCUMULATOR RIGHT

Octal Code 7010

Mnemonic CAR

The contents of ACCM and CRYRG together are circulated right by 1 bit position. The content of least significant bit is transferred to CRYRG and the content of CRYRG goes to the most significant bit of ACCM.

CIRCULATE ACCUMULATOR LEFT

Octal Code 7020

Mnemonic CAL

The contents of ACCM and CRYRG together are circulated left by one bit position. The content of most significant bit of ACCM goes to CRYRG and the content of CRYRG goes to least significant bit of ACCM.

CIRCULATE TWO RIGHT

Octal Code 7014

Mnemonic CTR

The contents of ACCM and CRYRG together are circulated right by two bit positions.

CIRCULATE TWO LEFT

Octal Code 7024

Mnemonic CTL

The contents of ACCM and CRYRG together are circulated left by two bit positions.

COMPLEMENT CARRY REGISTER

Octal Code 7040

Mnemonic CMC

The content of CRYRG is complemented.

COMPLEMENT ACCUMULATOR

Octal Code 7100

Mnemonic CMA

The contents of ACCM are set to one's complement of the current contents of the ACCM. Each bit of the ACCM is complemented.

INCREMENT ACCUMULATOR

Octal Code 7200

Mnemonic IAC

The content of the ACCM is incremented by one in twos complement arithmetic.

COMBINED INSTRUCTIONSCOMPLEMENT AND INCREMENT ACCUMULATOR

Octal Code 7300

Mnemonic CIA

The contents of the ACCM are converted into the twos complement number. (It is combination of CMA and IAC)

SET CARRY REGISTER

Octal Code 7042

Mnemonic STC

The CRYRG is set to contain a binary one. (It is combination of CLC and CMC).

SET ACCUMULATOR

Octal Code 7101

Mnemonic STA

Each bit of the ACCM is set to contain a binary 1. (It is combination of CIA and CMA).

(D) REGISTER SET-2

The micro-instruction format is shown in Figure-5. The primary micro-instructions are explained subsequently. Any logical combination of bits within this group can be combined into one micro-instruction.

If skips are combined in a single instruction, the inclusive OR of the conditions determines the skip when bit 9 is a zero; and the AND of the inverse of the conditions determines the skip when bit 9 is a one. If bit 9 is a zero and bits 6 and 8 are one, the next instruction is skipped if either the contents of the ACCM are minus or if the content of CRYRG are non-zero. If bit 9 is a one and bits 6 and 8 are one too, the next instruction is skipped if the contents of ACCM are positive and the CRYRG is zero.

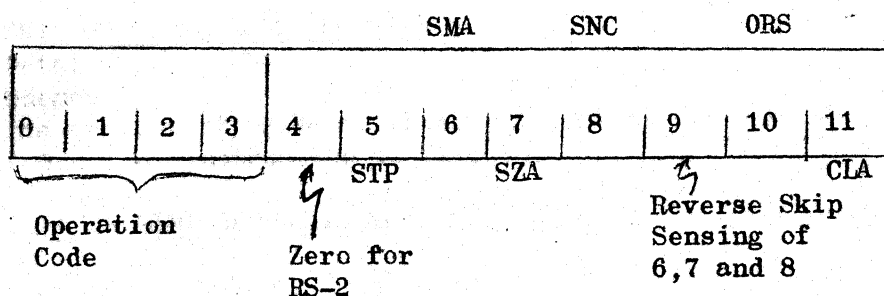


Figure-5 Format RS-2 Instruction Set

When micro-programmed the Logical Sequence in which the instructions are executed are:

1. (When bit 9 is a zero) Either SMA or SZA or SNC
2. (When bit 9 is a one) SPA and SNA and SZC
3. CLA
4. ORS, STP

CLEAR ACCUMULATOR

Octal Code 7401

Mnemonic CLA

Each bit of the ACCM is cleared to contain a binary zero.

OR WITH SWITCH REGISTER

Octal Code 7402

Mnemonic ORS

Inclusive OR function is performed between Switch Register and ACCM. When combined with CLA, the ORS performs a transfer of the contents of SWRG into the ACCM.

UNCONDITIONAL SKIP

Octal Code 7404

Mnemonic SKP

Next sequential instruction is skipped.

SKIP ON NON-ZERO CARRY REGISTER

Octal Code 7410

Mnemonic SNC

The contents of CRYRG are checked. If there is a 1 the next sequential instruction is skipped.

SKIP ON ZERO-CARRY REGISTER

Octal Code 7414

Mnemonic SZC

The content of CRYRG is checked, and if it contains a 0, the next sequential instruction is skipped.

SKIP ON ZERO-ACCUMULATOR

Octal Code 7420

Mnemonic SZA

Each bit of the ACCM is checked, and if each bit of ACCM contains a zero, the next instruction is skipped.

SKIP ON NON-ZERO ACCUMULATOR

Octal Code 7424

Mnemonic SNA

Each bit of the ACCM is checked, and if any bit or bits contain a 1, the next sequential instruction is skipped.

SKIP ON MINUS ACCUMULATOR

Octal Code 7440

Mnemonic SMA

The content of the most significant bit of the ACCM is checked, and if it contains a 1, indicating that the ACCM has a negative number, the next sequential instruction is skipped.

SKIP ON POSITIVE ACCUMULATOR

OCTAL CODE 7444

Mnemonic SPA

The content of the most significant bit of the ACCM is checked and if it contains a zero, indicating a positive number, the next sequential instruction is skipped.

STOP

Octal Code 7500

Mnemonic STP

This terminates the program. This command can be combined with other instructions of the RS-2 group, which are executed before program stops.

(E) INPUT OUTPUT INSTRUCTIONS

The operation code for input/output instructions is 64(octal).

The instruction format is given in Figure-6.

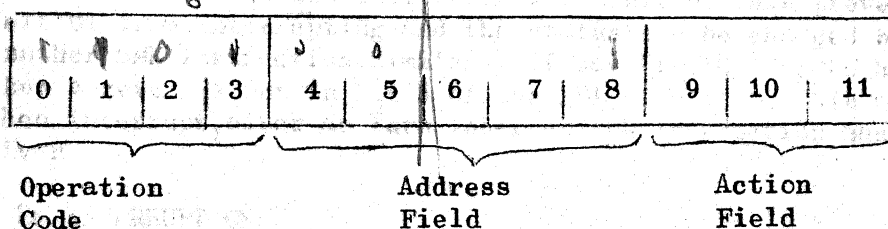


Figure-6 Format Input/Output Instruction Set

Bits 0 through 3 form the operation code; bits 4 through 8 form the Address field and bits 9 through 11 form the Action field.

The four types of I/O instructions may be divided as:

<u>ADDRESS FIELD</u>	<u>ACTION FIELD</u>	<u>TYPE OF INSTRUCTION</u>
Zero	Zero	Illegal
Zero	Non-Zero	Interrupt Status
Non-zero	Zero	Interrupt Sense
Non-Sense	Non-Zero	Device Attention

The Interrupt Sense Instruction occurs when one or more I/O devices are caused to interrupt.

INTERRUPT STATUS INSTRUCTIONS

These instructions are used for turning the Interrupt ON or OFF, and also for enabling only some of the devices to cause interrupt.

SET MASK FLIP FLOP

Octal Code 6401

Mnemonic SMK

By this only some of the I/O devices are enabled to interrupt. When this instruction occurs, then if bit 11 of the ACCM has a 1 then a 1 is put in bit 5 of the DDL of Unit 01. This prevents Unit 01 from interrupting and the status can be changed by giving another SMK instruction. Similarly if say bit 10 of ACCM has a zero then a zero is put in bit 5 of the DDL of Unit 02. This unit can then interrupt, after an Turn Interrupt ON instruction has been given

TURN INTERRUPT ON

Octal Code 6402

Mnemonic ION

It enables the computer to respond to an interrupt request. After this instruction is given the computer constantly checks for any interrupts after every instruction.

TURN INTERRUPT OFF

Octal Code 6404

Mnemonic IOF

It disables interrupt.

Octal Codes 6405, 6406 and 6407 are illegal instructions as they

represent conflicting operations. Octal Code 6403 is micro-programmed to give SMK and ION.

DEVICE ATTENTION INSTRUCTIONS

The address field contains the code of the selected device. The "device flag has 1" means that the device is ready. If the device flag has zero then the device is not ready.

UNIT 01 INPUT KEYBOARD READER

SENSE KEYBOARD

Octal Code 6414

Mnemonic KSF

Skip one instruction if device flag has 1.

CLEAR DEVICE

Octal Code 6412

Mnemonic KCC

Set zero in device flag and in accumulator.

OR ACCUMULATOR WITH BUFFER

OCTAL CODE 6411

Mnemonic KRS

Read one character from Keyboard. The contents of the device buffer are Inclusively ORed with bits 4 through 11 of ACCM and the result is placed in the ACCM. Bits 0 through 3 of ACCM are unchanged.

CLEAR AND .OR

Octal Code 6413

Mnemonic KRB

The instructions KCC and KRS are performed in sequence.

Octal Codes 6415, 6416 and 6417 are illegal instructions as they represent conflicting operations.

UNIT 02 INPUT PAPER TAPE READER

SENSE TAPE READER

Octal Code 6424

Mnemonic KSP

Skip one instruction if device flag has one.

CLEAR DEVICE

Octal Code 6422

Mnemonic KCS

Set zero in device flag and in ACCM.

OR ACCUMULATOR WITH BUFFER

Octal Code 6421

Mnemonic KRC

Read one character from paper tape. The contents of the device buffer are Inclusively ORed with bits 4 through 11 of ACCM and the result is placed in the ACCM. Bits 0 through 3 are unchanged, in the ACCM.

CLEAR AND OR

Octal Code 6423

Mnemonic KRP

The instructions KCS and KRC are performed in sequence.

Octal Codes 6425, 6426 and 6427 are illegal instructions as they represent conflicting operations.

UNIT 03 OUTPUT TELEPRINTERSENSE TELEPRINTER

Octal Code 6434

Mnemonic TSF

Skip one instruction if device flag has 1.

CLEAR DEVICE

Octal Code 6432

Mnemonic TCF

Set zero in device flag.

OR BUFFER WITH ACCUMULATOR

Octal Code 6431

Mnemonic TPC

Bits 4 through 11 of ACCM are placed in the device buffer and one character is printed out. Contents of ACCM are unchanged.

CLEAR AND OR

Octal Code 6433

Mnemonic TLS

The instructions TCF and TPC are performed in sequence.

Octal Codes 6435, 6436 and 6437 are illegal instructions as they represent conflicting operations.

UNIT 04OUTPUT PAPER TAPESENSE TAPE PUNCH

Octal Code 6444

Mnemonic TSP

Skip one instruction if device flag has 1.

CLEAR DEVICE

Octal Code 6442

Mnemonic TCP

Set zero in device flag.

OR BUFFER WITH ACCUMULATOR

Octal Code 6441

Mnemonic TPS

Bits 4 through 11 of ACCM are placed in the device buffer and one character is punched out. The contents of ACCM are unchanged.

CLEAR AND OR

Octal Code 6443

Mnemonic TLC

The instructions TCP and TPS are performed in sequence.

Octal Codes 6445, 6446 and 6447 are illegal instructions as they represent conflicting operations.

APPENDIX 2THE SIMULATOR SUBROUTINES

A brief description of how the various subroutines are used and called in the Simulator, is given.

CALL STRF

This includes the subprograms for all the storage-reference instructions. Depending on the particular instruction suitable entry is made into the subroutine, for execution of the instruction.

CALL RS1

This includes the subprograms for all the Register Set-1 instructions. Depending on how the instruction is microprogrammed suitable entry or entries are made in the subroutine for execution of the instruction.

CALL TCMP(X,Y)

The contents of location X in twos complement notation are converted to the sign and magnitude notation and stored in location Y. Contents of location X are unchanged unless Y is same as X.

CALL PACK(X,n)

The contents of location X are incremented by the amount n, and stored back in X in twos complement notation.

CALL READ(MAR,MDR)

The contents of location given in location MAR are copied into location MDR.

CALL

WRITE(MAR,MDR)

The contents of MDR are copied into the location given by the contents of MAR.

CALL

MMRY

This prints out the contents of the entire core-memory of TDC-12 together with all the electronic registers and the CLOCK.

CALL

DAMP

This is called from the MMRY subroutine when one or more than one blocks of 16 locations each, have the same contents. It prints out a brief message indicating the range of locations in TDC-12 which have the same contents.

CALL

CDRDER

This is called only when the redundancy check indicator is turned ON while reading. The computer pauses after giving a message of reading error on the IBM typewriter of IBM-7044. In that case the input information is to be read again.

APPENDIX 3THE MAJOR STATE GENERATOR

The TDC-12 computer operates in one of the four major control states during each machine cycle. One or more states are entered to execute an instruction, which are determined by the Major State Generator. Only one state exists at a time and all states except Interrupt, are determined by the programmed instruction being executed. The various states are described.

FETCH A new instruction is obtained when this instruction is entered. The contents of memory cell specified by the PAC are placed in MDR and the operation code (bits 0 through 3) of this instruction word are placed in IR. The contents of PAC are then incremented by one. If a single cycle instruction is fetched, the operations specified are performed in the last part of the Fetch cycle, and then the next state is Fetch for the next instruction. If a multiple cycle instruction is fetched, the succeeding control state is either DEFER or EXECUTE.

DEFER When bit 4 of a Storage Reference Instruction is 1, the defer state is entered to obtain 12 bit address of the operand from the address specified by bits 5 through 11 of the instruction. The state is called defer because access to the operand is deferred to the next memory cycle.

EXECUTE This state is established only when a storage reference instruction is executed. The content of memory location addressed is transferred to MDR and the operation, specified by the

contents of the IR is performed. During a JUMP instruction this state is not entered. During JUMP TO SUBROUTINE instruction this state occurs to write the contents of PAC into the Core Memory Address designated by the instruction and to transfer this address into the PAC to change program control.

INTERRUPT This state is established for a DATA Interrupt or Program Interrupt. The interrupt occurs only at the completion of the current instruction. The Data Interrupt allows information to be transferred between core memory and external device via MDR. When this transfer is complete, the program sequence is resumed, from the point of interrupt. The program interrupt causes the sequence to be altered. The contents of PAC are stored in location 0000, and program control is transferred to location 0001.

APPENDIX : 4USING THE SIMULATOR

The TDC-12 simulator has been run and tested. A listing of the simulator (with explanatory comment cards) together with a set of sample results is included with this report. The simulator was originally on cards but has now been put on Tape No: 808 in the Computer Centre at IIT, Kanpur. To store and execute a program the order of the control cards required is given.

\$JOB ,TIME ,PAGES ,NAME \$TDC-12 SIMULATOR.

* PLEASE MOUNT PAPER TAPE ON PRINTER.

\$PAUSE MOUNT TAPE NO: 808 ON UNIT 04.

\$IBJOB

16

\$IEDIT U04, SRCH

8

\$IBMAP SIM NODECK

\$ENTRY

The Header Card for program to be stored.

DATA SET

The program to be stored and executed.

It is punched on cards in either
BCD or Binary Form

If at any time during the execution of a program, information is to be placed in the SWRG of TDC-12, put IBM-7044 in manual mode. Enter the desired information through keys into location 30007₈ which is the simulated Switch Register. Put IBM-7044 in Automatic Mode and press START on the console.

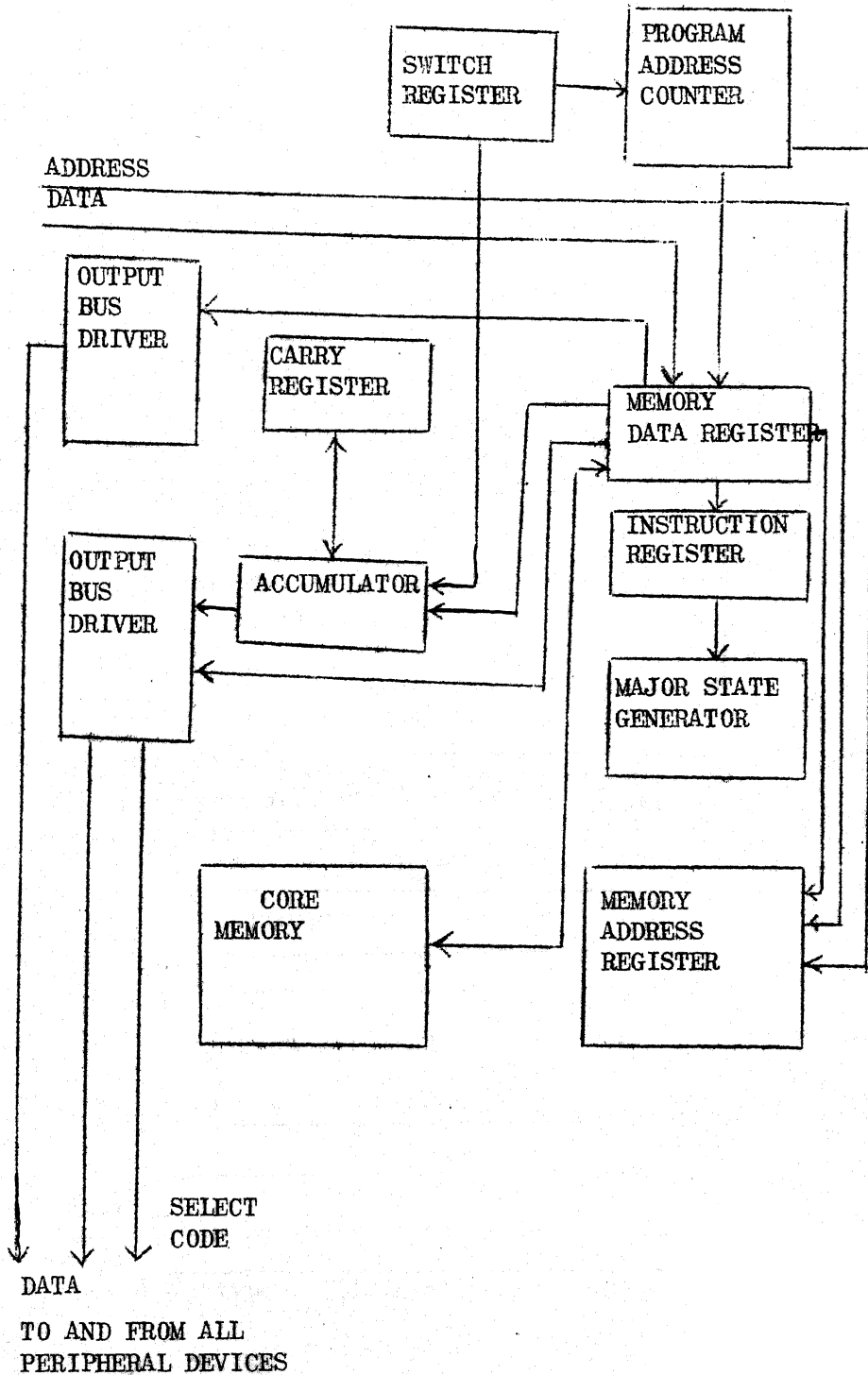


FIGURE-7 LOGICAL DESIGN OF TDC-12

Figure-8 Flow-Chart for the Simulator of TDC-12
On IBM-7044

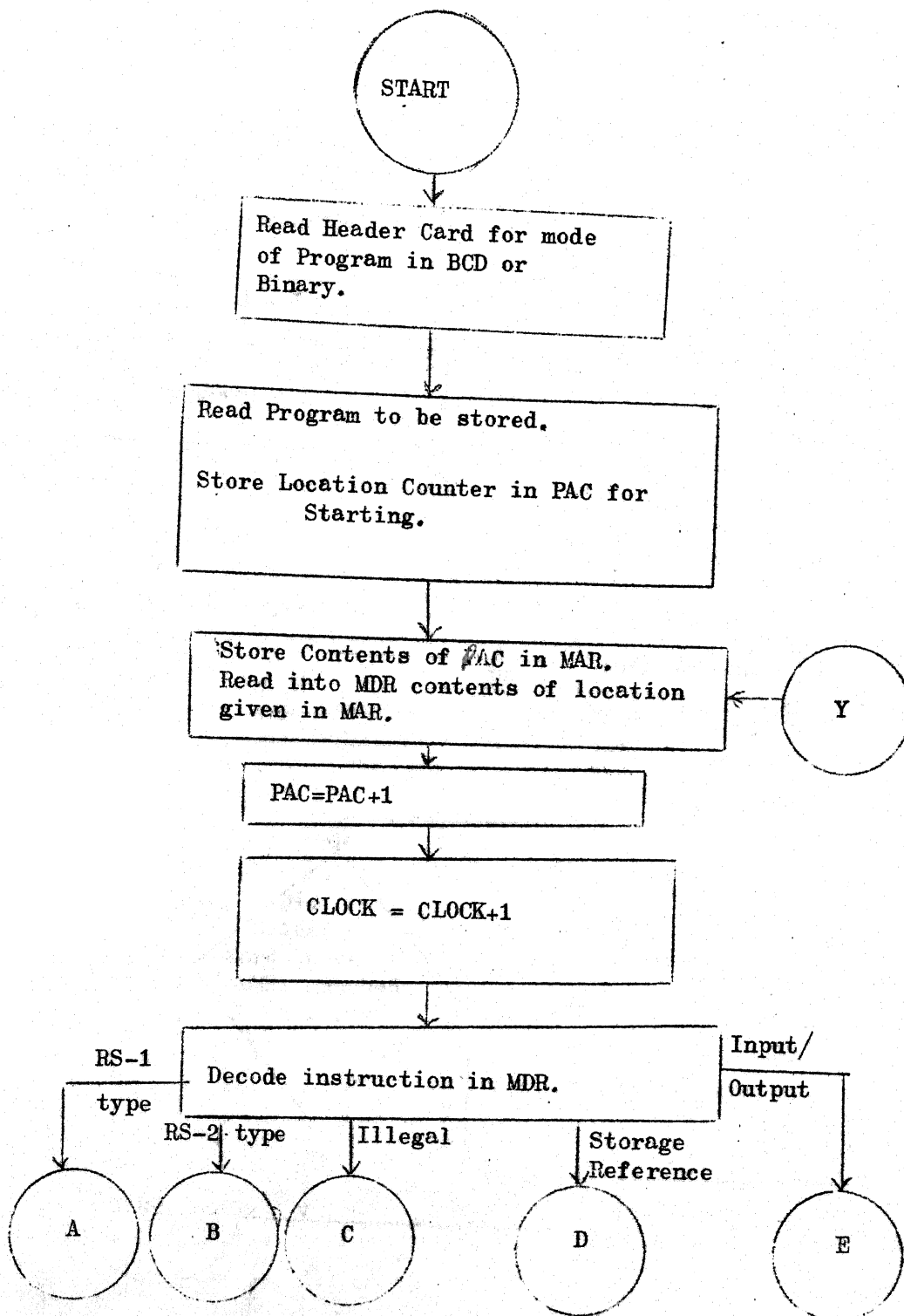


Figure-8 (Continued)

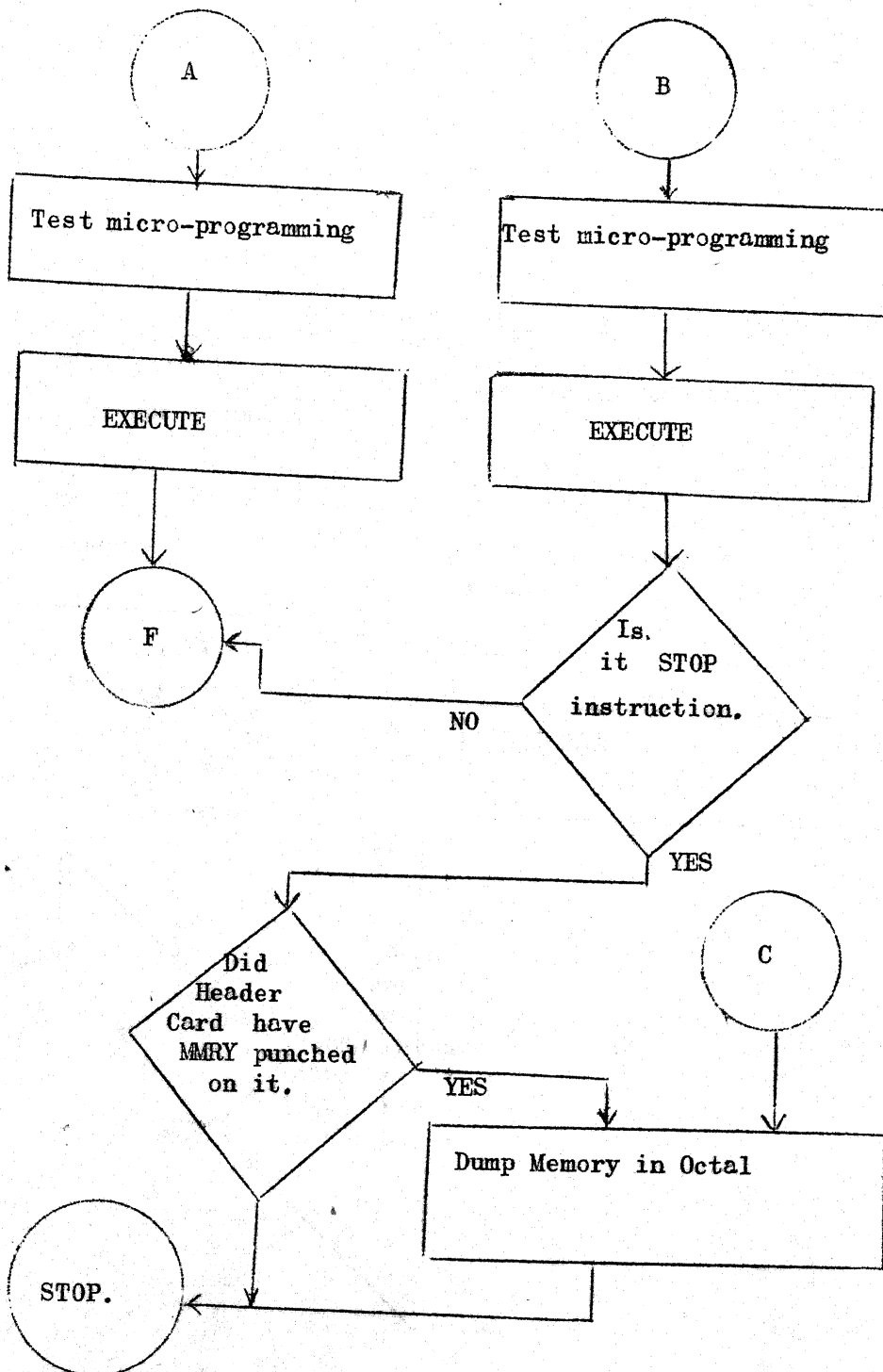


Figure-8 (Continued)

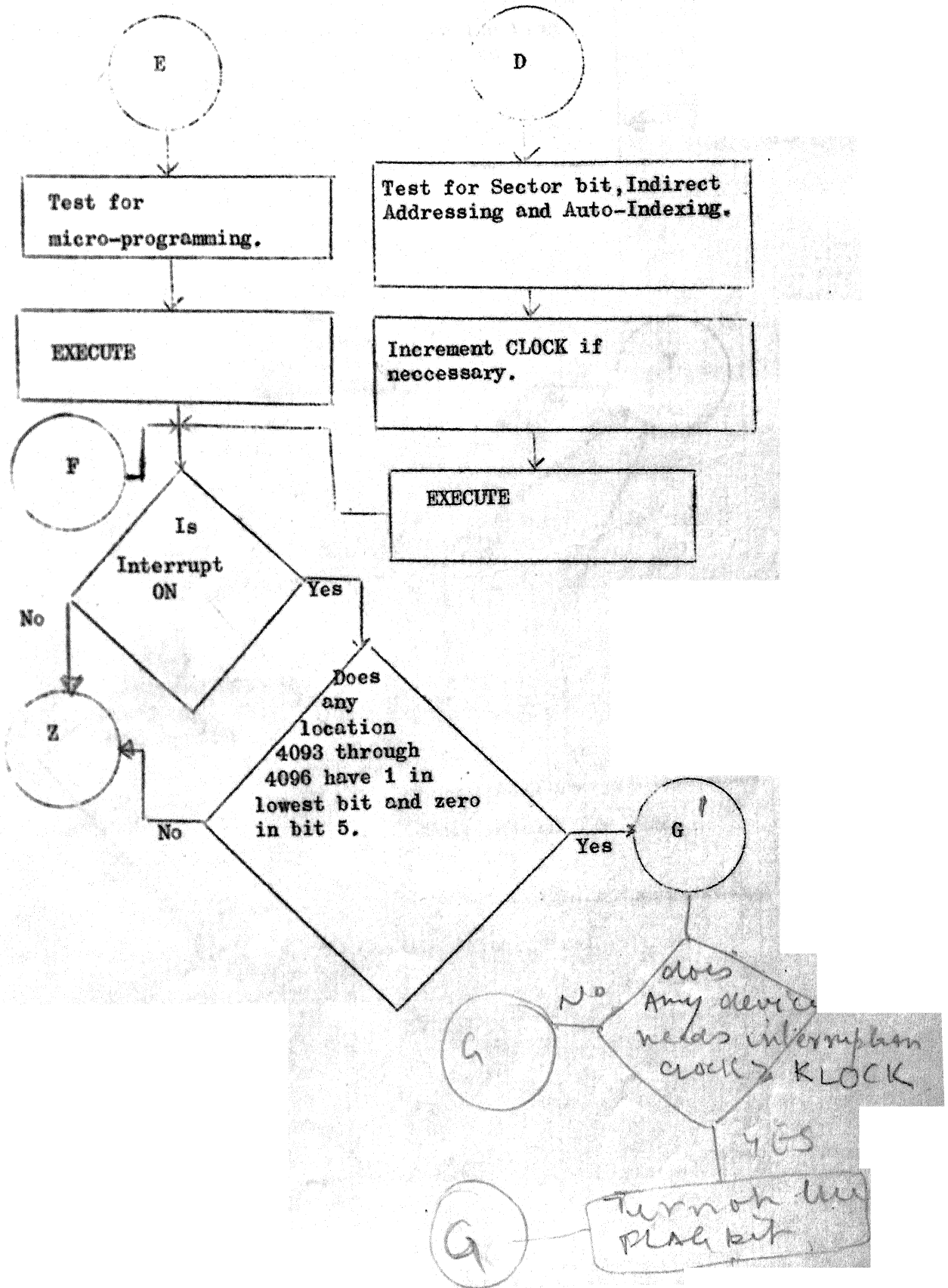


Figure-8 (Continued)

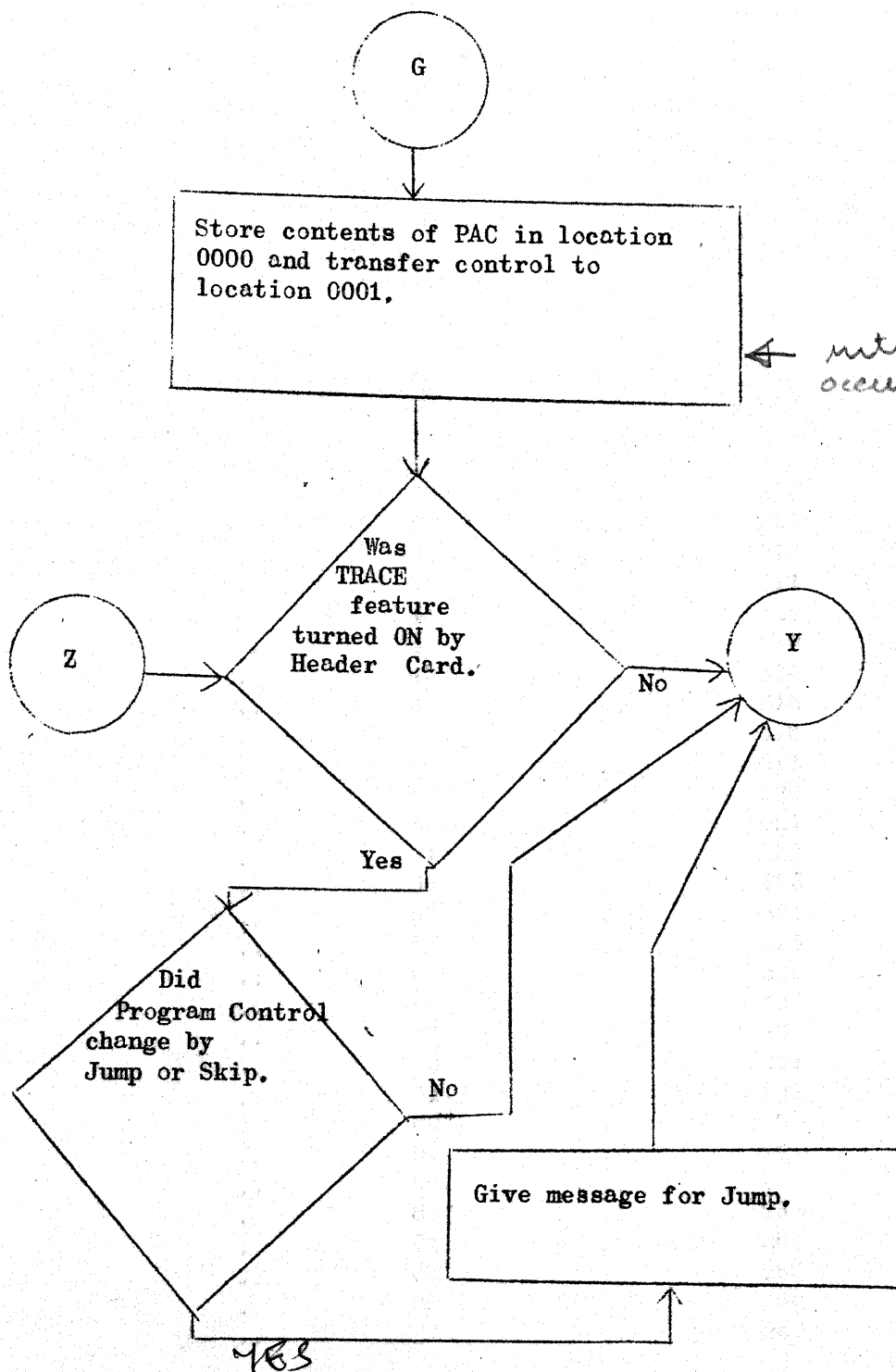


TABLE 1
 OCTAL CODES FOR I/O UNITS OF
 IBM-7044 and TDC-12

For TDC-12.....Model 33 ASR/KSR Teletype... ..as I/O Unit

For IBM-7044.....IBM-1402 and IBM-1403.....as I/O Unit

Character		6 bit code for IBM-7044		3 bit code for TDC-12
A	..	21	..	301
B	..	22	..	302
C	..	23	..	303
D	..	24	..	304
E	..	25	..	305
F	..	26	..	306
G	..	27	..	307
H	..	30	..	310
I	..	31	..	311
J	..	41	..	312
K	..	42	..	313
L	..	43	..	314
M	..	44	..	315
N	..	45	..	316
O	..	46	..	317
P	..	47	..	320
Q	..	50	..	321
R	..	51	..	322
S	..	62	..	323
T	..	63	..	324
U	..	64	..	325
V	..	65	..	326
W	..	66	..	327
X	..	67	..	330
Y	..	70	..	331
Z	..	71	..	332
0	..	00	..	260
1	..	01	..	261
2	..	02	..	262
3	..	03	..	263
4	..	04	..	264
5	..	05	..	265
6	..	06	..	266
7	..	07	..	267

TABLE 1 (Continued)

Character		6 bit code for IBM-7044		8 bit code for TDC-12
8	..	10	..	270
9	..	11	..	271
\$..	53	..	244
'(delimiter)	..	14	..	247
(..	74	..	250
)	..	34	..	251
*(Asterisk)	..	54	..	252
+(Plus)	..	20	..	253
, (Comma)	..	73	..	254
-(Minus)	..	40	..	255
.(Period)	..	33	..	256
/(Slash)	..	61	..	257
=(Equal to)	..	13	..	275
(Blank)	..	60	..	000
Line Feed	..	76	..	212
Carriage Return	..	77	..	215

LISTING .

```

£IBMAP SIM      ABSMOD
TTL  SIMULATOR OF TDC-12 ON IBM-7044
ORG      13000

```

*** MACROS FOR INTERRUPT*****

**** FOR 6404 TURN OFF INTERRUPT*****

```

IOF  MACRO  IOFQ
IOFQ MSP    INTR
      TRA    BODG
      ENDM   IOF

```

**** FOR 6402 TURN ON INTERRUPT***

```

ION  MACRO  IONQ
IONQ MSM    INTR
      TRA    BODH
      ENDM   ION

```

**** FOR 6401 SMK SET MASK FLIP FLOPS***

****BIT 11 IN ACCUMULATOR IF HAS 1 MASKS OUT UNIT 01 THAT IS
 ** IT PUTS MINUS IN LOCATION DEVICE+3.BIT 10 SIMILARLY HANDLES UNIT
 *** 02 AND SO ON.

```

SMK  MACRO  SMKB,SMKA,SMKQ
SMKQ AXT    4,2
      CLA    ACCM
      LGR    4
SMKB LGL    1
      LBT
      TRA    SMKA
      CLA    =1
      SAC    DEVICE+4,2,4
      TIX    SMKQ,2,1
      TRA    BEGIN
SMKA ZAC
      SAC    DEVICE+4,2,4
      TIX    SMKQ,2,1
      TRA    BEGIN
      ENDM   SMK

```

MASKED OUT

***** MACROS FOR INPUT*****

***** X COULD BE UNIT 1 OR 2**

*** FOR 64X4 KSF SKIP IF READY**

```

KSF  MACRO  I,KSFQ
KSFQ CLA    DEVIK+I
      LBT
      TRA    BODG
      CALL   PACK(PAC,1)
      TRA    BODG
      ENDM   KSF

```

***** 64X2 KCC CLEAR DEVIK AND ACCUMULATOR**

```

KCC  MACRO  I,KCCQ
KCCQ STZ    DEVIK+I
      STZ    ACCM

```

CLA	CLOCK		SIM00500
STO	KLOK+I		SIM00510
TRA	BUDH		SIM00520
ENDM	KCC		SIM00530
***	64X1 KRS	MOVE CHARACTER FROM BUFFER TO ACCUMJLATOR	SIM00540
**	IN DATA CARD RECORD MARK SHOWS END OF DATA, LATEST IT SHOULD BE IN		SIM00550
**	COLUMN 73. COL 77 AND 78 SPECIFY UNIT NO. BY PUTTING 01 OR 02		SIM00560
KRS	MACRO	Y, SETA, SETD, KRSQ, YY, REDCRD, ENDFIL, REDCHK, SETZ	SIM00570
KRSQ	ZAC		SIM00580
	PCS	BUFF&Y,,0	SIM00590
	SUB	RCMRK	SIM00600
	TNZ	SETA	SIM00610
**	MESSAGE	FOR DATA WANTED TO OPERATOR**	SIM00620
	ENB	=0	SIM00630
SETD	CLA	YY	SIM00640
	STO	SETB+4	SIM00650
REDCRD	SEN	664,,3	SIM00660
	RCHA	*+1	SIM00670
	IORD	SENSE,,1	SIM00680
	CAL	=00200000000000	SIM00690
	ANA	SENSE	SIM00700
	TNZ	ENDFIL	SIM00710
	PLT	SENSE IS READER READY	SIM00720
	TRA	ENDFIL	SIM00730
	TRCA	*+1 SWITCH OFF REDUNDANCY INDICATOR IF ON	SIM00740
	RDS	648,,3	SIM00750
	RCHA	*+1	SIM00760
	IORD	BUFF&Y,,13	SIM00770
	TRCA	REDCHK	SIM00780
	TRA	SETZ	SIM00790
ENDFIL	TSX	CORDER,1	SIM00800
	IORD	SETB,,6	SIM00810
	HPR		SIM00820
	TRA	REDCRD	SIM00830
REDCHK	TSX	CORDER,1	SIM00840
	IORD	RDREDR,,3	SIM00850
	HPR		SIM00860
	TRA	REDCRD	SIM00870
****	FIND IF DATA FED IS FOR DESIRED UNIT*****		SIM00880
SETZ	ENB	=0000003000003	SIM00890
	CAL	BUFF&Y+12	SIM00900
	LGR	3	SIM00910
	ARS	3	SIM00920
	LGR	3	SIM00930
	ZAC		SIM00940
	LGL	6	SIM00950
	SUB	=Y	SIM00960
	TNZ	REDCRD	SIM00970
	TRA	KRSQ	SIM00980
**	TRANSFER DATA FROM BUFFER TO ACCUMULATOR AFTER TABLE MATCH**		SIM00990
SETA	AXT	50,2	SIM01000
	ZAC		SIM01010
	PCS	BUFF&Y,,0	SIM01020
	CAS	IBM+50,2	SIM01030
	TIX	*-1,2,1	SIM01040

EEG018

UPDATE SUMMARY

TRA	++2		SIM01050
TIX	*-3,2,1		SIM01060
CAL	TELTYP+50,2		SIM01070
ORA	ACCM		SIM01080
SLW	ACCM		SIM01090
*** TO	SHIFT	CONTENT OF BUFFER BY 1 BYTE*(SIM01100
AXT	12,2		SIM01110
CAL	BUFF&Y+12,2		SIM01120
LDQ	BUFF&Y+13,2		SIM01130
LGL	6		SIM01140
SLW	BUFF&Y+12,2		SIM01150
TIX	*-4,2,1		SIM01160
TRA	BEGIN		SIM01170
ENDM	KRS		SIM01180
*****			SIM01190
***	MACROS	FOR OUTPUT*****	SIM01200
**	Y	STANDS FOR 3 OR 4 AS UNIT IS SELECTED**	SIM01210
**	FOR	64Y4 TSF SKIP IF UNIT READY	SIM01220
TSF	MACRO	I,TSFQ	SIM01230
TSFQ	CLA	DEVIK+I	SIM01240
	LBT		SIM01250
	TRA	BODG	SIM01260
	CALL	PACK(PAC,1)	SIM01270
	TRA	BODG	SIM01280
	ENDM	TSF	SIM01290
***	FOR	64Y2 TCF CLEAR DEVIK**	SIM01300
TCF	MACRO	J,TCFQ	SIM01310
TCFQ	STZ	DEVIK+J	SIM01320
	CLA	CLOCK	SIM01330
	STO	KLOK+J	SIM01340
	TRA	BODH	SIM01350
	ENDM	TCF	SIM01360
**	FOR	64Y1 TPC MOVE CHARACTER FROM ACCUMJLATOR TO BUFFER	SIM01370
****AND	OUTPUT	AT MOST 72 CHARACTERS PER LINE*****	SIM01380
TPC	MACRO	Y,TESZ,TESM,TESC,TESO,TESH,TESG,TESF,TESY,TESJ,TESX,TPCQ,	SIM01390
	ETC	TESL,TESK,TESN,TESV	SIM01400
TPCQ	CLA	ACCM	SIM01410
	LGR	8	SIM01420
	ZAC		SIM01430
	LGL	8	SIM01440
	STO	SAVED	SIM01450
	CAS	=0212	SIM01460
	TRA	++2	SIM01470
	TRA	TESM	SIM01480
	CAS	=0215	SIM01490
	TRA	TESC	SIM01500
	TRA	TESO	SIM01510
	TRA	TESC	SIM01520
TESM	STO	LINFID	SIM01530
	TRA	++2	SIM01540
TESO	STO	CARRET	SIM01550
	CLA	LINFID	SIM01560
	SUB	=0212	SIM01570
	TNZ	TESC	SIM01580
	CLA	CARRET	SIM01590

	SUB	=0215	SIM01600
	TZE	TESL	SIM01610
***	TO	MATCH THE TWO CODES***	SIM01620
TESC	AXT	48,1	SIM01630
	CLA	SAVED	SIM01640
	CAS	TELTYP+48,1	SIM01650
	TIX	*-1,1,1	SIM01660
	TRA	*+2	SIM01670
	TIX	*-3,1,1	SIM01680
**	TO	STORE THE CODE**	SIM01690
	CLA	IBM+48,1	SIM01700
	STO	HARSHA	SIM01710
**	TO	FIND WHICH BYTE HAS RM**	SIM01720
	AXT	12,1	SIM01730
TESH	AXT	6,2	SIM01740
	LDQ	BUFF&Y+12,1	SIM01750
TESG	LGL	6	SIM01760
	CCS	RCMRK,,5	SIM01770
	TRA	*+2	SIM01780
	TRA	TESF	SIM01790
	TIX	TESG,2,1	SIM01800
	TIX	TESH,1,1	SIM01810
***	MATCH	FO RM FOUND*****	SIM01820
TESF	PCS	HARSHA,,5	SIM01830
	TNX	TESJ,2,1	SIM01840
	LGL	6	SIM01850
	PCS	RCMRK,,5	SIM01860
	TNX	TESY,2,1	SIM01870
	LGL	6	SIM01880
	TRA	*-2	SIM01890
TESY	SLW	BUFF&Y+12,1	SIM01900
	TRA	TESX	SIM01910
**	WHEN RM	FOUND WAS IN LAST BYTE OF ONE WORD TO STORE IN 0 BYTE	SIM01920
**	OF NEXT	WORD**	SIM01930
TESJ	SLW	BUFF&Y+12,1	SIM01940
	LDQ	BUFF&Y+13,1	SIM01950
	LGL	6	SIM01960
	PCS	RCMRK,,5	SIM01970
	LGR	6	SIM01980
	STQ	BUFF&Y+13,1	SIM01990
**	TEST	IF BUFFER IS FULL. IF SO PRINT**	SIM02000
TESX	PCS	BUFF&Y+12,,0	SIM02010
	CCS	RCMRK,,5	SIM02020
	TRA	BEGIN	SIM02030
	TRA	TESZ	SIM02040
	TRA	BEGIN	SIM02050
TESL	AXT	12,1	SIM02060
TESK	AXT	6,2	SIM02070
	LDQ	BUFF&Y+12,1	SIM02080
TESN	LGL	6	SIM02090
	CCS	RCMRK,,5	SIM02100
	TRA	*+2	SIM02110
	TRA	*+3	SIM02120
	TIX	TESN,2,1	SIM02130
	TIX	TESK,1,1	SIM02140

```

PCS      BLANK,,5
TNX      TESV,2,1
LGL      6
TRA      *-2
TESV     SLW      BUFF&Y+12,1
*** PRINT STATEMENT*****
TESZ     STZ      LINFID
          STZ      CARRET
          CLA      BLANK
          SAC      BUFF&Y+12,,0
          WRS      650,,3
          RCHA     **1
          IORD     BUFF&Y,,15
*** INITIALIZE BUFFER TO BLANKS AND A RM IN 0 BYTE OF
** FIRST BUFFER WORD*****
          AXT      13,1
          CLA      BLANK
          STO      BUFF&Y+13,1
          TIX      *-1,1,1
          PCS      RCMRK,,5
          SAC      BUFF&Y,,0
          TRA      BEGIN
          ENDM     TPC
          EJECT
*****
*** TO INITIALIZE TDC MEMORY*****
AARMBH AXT      4096,4
          STZ      12288,4
          TIX      *-1,4,1
***** TO CHOOSE BCD OR BINARY FORM OF READING*****
** THE FIRST CARD HAS COL 3 THRU 6 BCDM OR BNRY FOR FORM OF READ
** COL 9 THRU 12 TRAS FOR TRACE FEATURE
**** COL 15 THRU 18 FOR MEMORY DUMP MMRY ON FULL COMPLETION
          RDS      648,,3
          RCHA     **1
          IORD     BAFF,,3
          ENB      =0
          TRCA     **1 SWITCH OFF REDUNDANCY CHECK
          CLA      BAFF
          SUB      =H BNRY
          TZE      PARHO
          CLA      BAFF
          SUB      =H BCDM
          TZE      START
          TRA      CHAPEL
***** READ IN TEST PROGRAM *****
**** TO READ IN BINARY CARDS.COLUMN 1 HAS 7-9 PUNCH,COLUMN 2 HAS
* NUMBER OF TDC INSTRUCTIONS, COLUMN 3 HAS STARTING LOCATION
** FROM WHERE INSTRUCTIONS ARE LOADED IN TDC MEMORY
PARHO RDS      664,,3
          RCHA     **1
          IORD     BUFF,,27
          TRCA     BNRYER
* TO SPLIT AND STORE AS TDC WORDS
          AXT      82,2

```

```

SIM02150
SIM02160
SIM02170
SIM02180
SIM02190
SIM02200
SIM02210
SIM02220
SIM02230
SIM02240
SIM02250
SIM02260
SIM02270
SIM02280
SIM02290
SIM02300
SIM02310
SIM02320
SIM02330
SIM02340
SIM02350
SIM02360
SIM02370
SIM02380
SIM02390
SIM02400
SIM02410
SIM02420
SIM02430
SIM02440
SIM02450
SIM02460
SIM02470
SIM02480
SIM02490
SIM02500
SIM02510
SIM02520
SIM02530
SIM02540
SIM02550
SIM02560
SIM02570
SIM02580
SIM02590
SIM02600
SIM02610
SIM02620
SIM02630
SIM02640
SIM02650
SIM02660
SIM02670
SIM02680
SIM02690

```


	AXT	28,1	SIM02700
BINC	AXT	3,4	SIM02710
	TXI	*+1,1,-1	SIM02720
	TXI	*+1,2,-1	SIM02730
	LDQ	BUFF+27,1	SIM02740
BINB	ZAC		SIM02750
	LGL	12	SIM02760
	STO	ETONE+81,2	SIM02770
	TNX	BINC,4,1	SIM02780
	TIX	BINB,2,1	SIM02790
***	THE	BREAK IS OVER.CHECK FOR LEGALITY OF INPUT	SIM02800
	CLA	ETONE	SIM02810
	SUB	=05	SIM02820
	TNZ	BIND	SIM02830
*	TO PARK	INTO TDC MEMORY	SIM02840
	CLA	ETONE+2	SIM02850
	ORA	BAAM	SIM02860
	STO	CUFF	SIM02870
	LXA	ETONE+1,1	SIM02880
	CLA	DRAKE	SIM02890
	ADD	ETONE+1	SIM02900
	STA	*+1	SIM02910
BINE	CLA	** ,1	SIM02920
	STO*	CUFF	SIM02930
	CLA	CUFF	SIM02940
	ADD	=1	SIM02950
	STO	CUFF	SIM02960
	TIX	BINE,1,1	SIM02970
	TRA	PARHO	SIM02980
***	TEST	FOR END CARD WHICH HAS A 12 7 9 PUNCH**	SIM02990
BIND	CLA	ETONE	SIM03000
	SUB	=04005	SIM03010
	TZE	BINF	SIM03020
CHAPEL	WRS	650,,3	SIM03030
	RCHA	*+1	SIM03040
	IORD	BING,,5	SIM03050
	TRA	BOBJJ	SIM03060
BNRYER	TSX	CDRDER,1	SIM03070
	IORD	RDREDR,,3	SIM03080
	HPR		SIM03090
	TRA	PARHO	SIM03100
***	IN	END CARD COLUMN 2 HAS PROGRAM STARTING LOCATOION	SIM03110
BINF	CLA	ETONE+1	SIM03120
	TRA	BEG+1	SIM03130
***	IN	BCD MODE COL 9 THRU 12 HAVE DATA	SIM03140
***	AND	COL 3 THRU 6 HAVE LOCATION	SIM03150
START	RDS	648,,3	SIM03160
	RCHA	*+1	SIM03170
	IORD	BUFF,,13	SIM03180
	TRCA	BCDMER	SIM03190
	WRS	650,,3	SIM03200
	RCHA	*+1	SIM03210
	IORD	BUFF,,13	SIM03220
	AXT	2,4	SIM03230
HOBB	AXT	4,2	SIM03240

EEG018

UPDATE SUMMARY

HOB	CLA	BUFF+2,4	SIM03250
	LGR	3	SIM03260
	ARS	3	SIM03270
	TIX	HOB,2,1	SIM03280
	ZAC		SIM03290
	LGL	12	SIM03300
	STO	CUFF+2,4	SIM03310
	TIX	HOB,4,1	SIM03320
	CLA	BUFF	SIM03330
	SUB	=H DEND	SIM03340
	TZE	BEG	SIM03350
	CLA	CUFF	SIM03360
	ORA	BAAM	SIM03370
	STO	CUFF	SIM03380
	CLA	CUFF+1	SIM03390
	STO*	CUFF	SIM03400
	TRA	START	SIM03410
BCDMER	TSX	CDRDER,1	SIM03420
	IORD	RDREDR,,3	SIM03430
	HPR		SIM03440
	TRA	START	SIM03450
BEG	CLA	CUFF+1	SIM03460
	STO	SWRG	SIM03470
	STO	PACA	SIM03480
	STO	PAC	SIM03490
	ENB	=0000003000003	SIM03500
	EJECT		SIM03510
*****			SIM03520
	REM	TDC 12 SIMULATOR	SIM03530
*****			SIM03540
*****			SIM03550
*** TEST FOR JUMP IN PROGRAM*****			SIM03560
**** IN CASE OF JUMP IN TDC MESSAGE IS GIVEN AS TRACE FEATURE			SIM03570
BEGINN	CLA	BAFF+1	SIM03580
	SUB	=H TRAS	SIM03590
	TNZ	PEN	SIM03600
	CLA	PAC	SIM03610
	SUB	PACA	SIM03620
	TZE	PEN	SIM03630
	CLA	PACA	SIM03640
	SUB	=1	SIM03650
	STO	PACA	SIM03660
	AXT	4,2	SIM03670
NEEF	AXT	4,4	SIM03680
	LDQ	PACA+4,2	SIM03690
	LGL	24	SIM03700
	ZAC		SIM03710
NIFF	ALS	3	SIM03720
	LGL	3	SIM03730
	TIX	NIFF,4,1	SIM03740
	ALS	6	SIM03750
	PCS	BLANK,,5	SIM03760
	ALS	6	SIM03770
	PCS	BLANK,,5	SIM03780
	SLW	GUMP+8,2	SIM03790

TIX	NEEF,2,2		SIM03800
WRS	650,,3		SIM03810
RCHA	*+1		SIM03820
IORD	GUMP,,7		SIM03830
* SET MAJOR STATE GENERATOR TO FETCH MODE			SIM03840
PEN STZ	MSG		SIM03850
**** PSEUDO CLOCK FEATURE*****			SIM03860
CLA	CLOCK		SIM03870
ADD	=1 ONE CYCLE		SIM03880
STO	CLOCK		SIM03890
* LOCATION OF INSTRUCTION TO BE EXECUTED IS IN PAC			SIM03900
UTHANT EQU	PAC		SIM03910
CLA	PAC		SIM03920
ORA	BAAM		SIM03930
STO	MAR		SIM03940
CALL	READ(MAR,MDR)		SIM03950
* INSTRUCTION TO BE EXECUTED IS NOW IN MDR.TO DECODE IT.			SIM03960
* ALSO INCREMENT PAC			SIM03970
CALL	PACK(PAC,1)		SIM03980
***** FACILITY FOR USING TRACE FEATURE*****			SIM03990
CLA	PAC		SIM04000
STO	PACA		SIM04010
* PLACE OP CODE IN INSTRUCTION REGISTER IR			SIM04020
BOAR CLA	MDR		SIM04030
LGR	8		SIM04040
STO	IR		SIM04050
* * TO SEE IF IT IS STORAGE REFERENCE TYPE OR NOT			SIM04060
CLA	=13		SIM04070
CCS	IR,,5		SIM04080
TRA	DCSR	TO HANDLE STORAGE REFERENCE	SIM04090
TRA	DCIO	INPUT OUTPUT TYPE HANDLE HERE	SIM04100
CLA	=15		SIM04110
CCS	IR,,5		SIM04120
TRA	DCRS1	HANDLE RS 1 INSTRUCTIONS	SIM04130
TRA	DCRS2	HANDLE RS 2 INSTRUCTIONS THERE	SIM04140
*****			SIM04150
EJECT			SIM04160
*** FOR I/O INSTRUCTIONS ENTER HERE**			SIM04170
** ARE BITS 9,10 11 ZERO			SIM04180
DCIO LDQ	MDR		SIM04190
LGL	28		SIM04200
ZAC			SIM04210
LGL	5		SIM04220
STO	UNIT	BITS 4 THRU 8 STORED**	SIM04230
ZAC			SIM04240
LGL	3		SIM04250
STO	FLIP	BITS 9 THRU 11 STORED	SIM04260
TNZ	B0DB		SIM04270
** WHEN BITS 9,10 11 ARE ZERO ARE BITS 4 THRU 8 ZERO*			SIM04280
BODA CLA	UNIT		SIM04290
TZE	ERROR		SIM04300
MIT	INTR	IS INTERRUPT ON	SIM04310
TRA	BEGIN		SIM04320
***** SET MSG TO INTERRUPT MODE**			SIM04330
CLA	=03		SIM04340


```

      STO      MSG
      CLA      PAC
      STO      8192
      CLA      =1
      STO      PAC
      MSP      INTR  TURN OF  INTERRUPT
      TRA      BEGIN
** FLIP DOES NOT HAVE ZERO.IS DEVICE ADDRESSED MORE THAN 4***
BODB  CLA      =4
      CAS      FLIP
      TRA      BODC
      TRA      BODC
      TRA      ERROR  ILLEGAL  IMSTRUCTION
BODC  CAS      UNIT
      TRA      BODD
      TRA      BODD
      TRA      ERROR  PRESENTLY THERE ARE ONLY 4 I/O UNITS
** STORE CONTENTS OF UNIT IN XR4
BODD  LXA      UNIT,4
** TEST FOR BITS 9,10 11 FOR DECODING  INSTRUCTION*****
      CLA      FLIP
      LGR      2
      LBT      TEST  FOR 9  BIT
      TRA      BODG
      TRA*     CHAR+4,4
BODG  CLA      FLIP
      LGR      1
      LBT      TEST  FOR 10  BIT
      TRA      BODH
      TRA*     DO+4,4
BODH  CLA      FLIP
      LBT      TEST  FOR 1  BIT
      TRA      BEGIN
      TRA*     EK+4,4
      EJECT
*****
**** CALLING OF MACROS FOR I/O****
*** FOR INTERRUPT
      PMC      ON
      IOF      IOFQ
      ION      IONQ
      SMK      SMKB,SMKA,SMKQ
*** FOR UNIT 1  KEYBOARD  READER***
      KSF      3,KSFQ
      KCC      3,KCCQ
      KRS      01,SETA,SETD,KRSQ(=H  01)REDCRD,ENDFIL,REDCHK,SETZ
| *** FOR UNIT 2  PAPER TAPE READER*****
      KSF      2,KSFQA
      KCC      2,KCCQA
      KRS      02,SETAA,SETDA,KRSQA(=H  02)REDCR,ENDFI,REDCH,SETZA
**** FOR UNIT 3  TELETYPE  PRINTER*****
      TSF      1,TSFQ
      TCF      1,TCFQ
      TPC      03,TESZ,TESM,TESC,TESO,TESH,TESG,TESF,TESY,TESJ,TESX,
      ETC      TPCQ,TESL,TESK,TESN,TESV

```

SIM04350
 SIM04360
 SIM04370
 SIM04380
 SIM04390
 SIM04400
 SIM04410
 SIM04420
 SIM04430
 SIM04440
 SIM04450
 SIM04460
 SIM04470
 SIM04480
 SIM04490
 SIM04500
 SIM04510
 SIM04520
 SIM04530
 SIM04540
 SIM04550
 SIM04560
 SIM04570
 SIM04580
 SIM04590
 SIM04600
 SIM04610
 SIM04620
 SIM04630
 SIM04640
 SIM04650
 SIM04660
 SIM04670
 SIM04680
 SIM04690
 SIM04700
 SIM04710
 SIM04720
 SIM04730
 SIM04740
 SIM04750
 SIM04760
 SIM04770
 SIM04780
 SIM04790
 SIM04800
 SIM04810
 SIM04820
 SIM04830
 SIM04840
 SIM04850
 SIM04860
 SIM04870
 SIM04880
 SIM04890

```

*** FOR UNIT 4 PAPER TAPE PUNCH*****
    TSF      0,TSFQA
    TCF      0,TCFQA
    TPC      04,TESZA,TESMA,TESCA,TESQA,TESHA,TESGA,TESFA,
    ETC      TESYA,TESJA,TESXA,TPCQA,TESLA,TESKA,TESNA,TESVA
    PMC      OFF
*****
EJECT
*****
**** ROUTINE MESSAGES ON IBM TYPEWRITER FOR OPERATOR ACTION
***   LIKE CARD ERROR WHILE READING
***** CALLED BY TSX CORDER,1
CORDER WRS      512,,4
    RCHA      1,1
    TRA      2,1
    EJECT
*****
* FOR RS 1 INSTRUCTIONS ENTER HERE
DCRS1 CLA      MDR
**** TEST FOR NOP*****
    SUB      =07000
    TNZ      CDZ
    AXT      8,1
    CALL     RS1
    TRA      BEGIN
CDZ CLA      MDR
** TEST FOR CLA 7001
    LBT
    TRA      CDA
    AXT      7,1
    CALL     RS1
** TEST FOR CLC 7002
CDA LGR      1
    LBT
    TRA      CDB
    AXT      6,1
    CALL     RS1
** TEST FOR CMA 7100
CDB LGR      5
    LBT
    TRA      CDC
    AXT      2,1
    CALL     RS1
**** TEST FOR CMC 7040
CDC LGL      1
    LBT
    TRA      CDD
    AXT      3,1
    CALL     RS1
*** TEST FOR IAC 7200
CDD LGR      2
    LBT
    TRA      CDE
    LGL      2
    ZAC
    WITH IAC NO ROTATE

```

SIM04900
 SIM04910
 SIM04920
 SIM04930
 SIM04940
 SIM04950
 SIM04960
 SIM04970
 SIM04980
 SIM04990
 SIM05000
 SIM05010
 SIM05020
 SIM05030
 SIM05040
 SIM05050
 SIM05060
 SIM05070
 SIM05080
 SIM05090
 SIM05100
 SIM05110
 SIM05120
 SIM05130
 SIM05140
 SIM05150
 SIM05160
 SIM05170
 SIM05180
 SIM05190
 SIM05200
 SIM05210
 SIM05220
 SIM05230
 SIM05240
 SIM05250
 SIM05260
 SIM05270
 SIM05280
 SIM05290
 SIM05300
 SIM05310
 SIM05320
 SIM05330
 SIM05340
 SIM05350
 SIM05360
 SIM05370
 SIM05380
 SIM05390
 SIM05400
 SIM05410
 SIM05420
 SIM05430
 SIM05440

LGL	3	INSTRUCTIONS	ALLOWED	SIM05450
TNZ	ERROR			SIM05460
AXT	1,1			SIM05470
CALL	RS1			SIM05480
TRA	BEGIN			SIM05490
*** TEST FOR CAR 7010***				SIM05500
CDE	LGL	4		SIM05510
	LBT			SIM05520
	TRA	CDF	NO CAR	SIM05530
	LGR	1		SIM05540
	LBT			SIM05550
	TRA	**+2	WITH CAR NO CAL ALLOWED	SIM05560
	TRA	ERROR		SIM05570
	AXT	5,1		SIM05580
	CALL	RS1		SIM05590
*** TEST FOR CTR 7014				SIM05600
	LGL	2		SIM05610
	LBT			SIM05620
	TRA	BEGIN		SIM05630
	CALL	RS1		SIM05640
	TRA	BEGIN		SIM05650
**** TEST FOR CAL 7020				SIM05660
CDF	LGR	1		SIM05670
	LBT			SIM05680
	TRA	CDG	NO CAL HERE	SIM05690
	AXT	4,1		SIM05700
	CALL	RS1		SIM05710
**** TEST FOR CTL 7024				SIM05720
	LGL	2		SIM05730
	LBT			SIM05740
	TRA	BEGIN		SIM05750
	CALL	RS1		SIM05760
	TRA	BEGIN		SIM05770
** TEST FOR 9 BIT PRESENT***				SIM05780
CDG	LGL	2		SIM05790
	LBT			SIM05800
	TRA	BEGIN		SIM05810
	TRA	ERROR	ONLY 9 BIT PRESENT	SIM05820
* THIS BLOCK CONTAINS THE SUBPROGRAMS FOR REGISTER SET 1 INSTRUCTIONS				SIM05830
RS1	SAVE	1,2,4		SIM05840
	SLW	VAS		SIM05850
	STQ	VAS+1	ACCM AND MQ SAVED	SIM05860
	TRA*	STALIN+8,1		SIM05870
STALIN	PZE	SUBA		SIM05880
	PZE	SUBB		SIM05890
	PZE	SUBC		SIM05900
	PZE	SUBD		SIM05910
	PZE	SUBE		SIM05920
	PZE	SUBH		SIM05930
	PZE	SUBI		SIM05940
	PZE	SUBK		SIM05950
* 7000	NOP	NO OPERATION		SIM05960
SUBA	TRA	LENIN		SIM05970
* 7001	CLA	CLEAR ACCUMULATOR		SIM05980
SUBB	STZ	UTHANT+3		SIM05990

	TRA	LENIN	SIM06000
* SUBC	7002	CLC CLEAR CARRY REGISTER	SIM06010
	STZ	UTHANT+4	SIM06020
	TRA	LENIN	SIM06030
* SUBD	7010	CAR CIRCULATE ACCUMULATOR RIGHT	SIM06040
	LDQ	ACCM	SIM06050
	LGL	24	SIM06060
	CLA	CRYRG	SIM06070
	LGR	1	SIM06080
	ZAC		SIM06090
	LGL	12	SIM06100
	STO	ACCM	SIM06110
	RQL	1	SIM06120
	STQ	CRYRG	SIM06130
	TRA	LENIN	SIM06140
* SUBE	7020	CAL CIRCULATE ACCUMULATOR LEFT	SIM06150
	CAL	ACCM	SIM06160
	LDQ	CRYRG	SIM06170
	RQL	35	SIM06180
	LGR	12	SIM06190
	ZAC		SIM06200
	LGL	1	SIM06210
	STO	CRYRG	SIM06220
	ZAC		SIM06230
	LGL	12	SIM06240
	STO	ACCM	SIM06250
	TRA	LENIN	SIM06260
* SUBH	7040	CMC COMPLEMENT CARRY REGISTER	SIM06270
	CAL	UTHANT+4	SIM06280
	COM		SIM06290
	LGR	1	SIM06300
	ZAC		SIM06310
	LGL	1	SIM06320
	SLW	UTHANT+4	SIM06330
	TRA	LENIN	SIM06340
* SUBI	7100	CMA COMPLEMENT ACCUMULATOR	SIM06350
	CAL	UTHANT+3	SIM06360
	COM		SIM06370
	LGR	12	SIM06380
	ZAC		SIM06390
	LGL	12	SIM06400
	SLW	UTHANT+3	SIM06410
	TRA	LENIN	SIM06420
* SUBK	7200	IAC INCREMENT ACCUMULATOR	SIM06430
	CAL	UTHANT+3	SIM06440
	ADD	=1	SIM06450
	LGR	13	SIM06460
	ZAC		SIM06470
	LGL	1	SIM06480
	STO	UTHANT+4	SIM06490
	ZAC		SIM06500
	LGL	12	SIM06510
	SLW	UTHANT+3	SIM06520
	TRA	LENIN	SIM06530
LENIN	CAL	VAS	SIM06540

EEG018

UPDATE SUMMARY

LDQ VAS+1 ACCM AND MQ RESTORED
 RETURN RS1
 EJECT

SIM06550

SIM06560

SIM06570

SIM06580

SIM06590

SIM06600

SIM06610

SIM06620

SIM06630

SIM06640

SIM06650

SIM06660

SIM06670

SIM06680

SIM06690

SIM06700

SIM06710

SIM06720

SIM06730

SIM06740

SIM06750

SIM06760

SIM06770

SIM06780

SIM06790

SIM06800

SIM06810

SIM06820

SIM06830

SIM06840

SIM06850

SIM06860

SIM06870

SIM06880

SIM06890

SIM06900

SIM06910

SIM06920

SIM06930

SIM06940

SIM06950

SIM06960

SIM06970

SIM06980

SIM06990

SIM07000

SIM07010

SIM07020

SIM07030

SIM07040

SIM07050

SIM07060

SIM07070

SIM07080

SIM07090

 * FOR RS 2 INSTRUCTION ENTER HERE
 ** BEGIN DECODING AND EXECUTION OF RS 2 SET OF INSTRUCTIONS HERE**

DCRS2 CAL MDR
 SUB =07400
 TZE ERROR
 CAL MDR
 ARS 2
 LBT TEST FOR 9 BIT

TRA NRA
 TRA NRI
 NRA ARS 1
 LBT TEST FOR 8 BIT
 TRA NRC

*** DOES SNC 7410 CONDITION HOLD**

NRB CAL CRYRG

LBT
 TRA NRC
 TRA NRD
 NRC CAL MDR
 ARS 4
 LBT TEST FOR 7 BIT
 TRA NRF

** DOES SZA 7420 CONDITION HOLD**

NRE CAL ACCM
 LGR 12

ZAC
 LGL 12
 TZE NRD
 NRF CAL MDR
 ARS 5

LBT TEST FOR 6 BIT
 TRA NRH

TEST FOR SMA 7440 CONDITION HOLDING

NRG CAL ACCM
 LGR 11

LBT
 TRA NRH
 TRA NRD

*** WHEN 9 BIT IS A 1 COME HERE**

NRI ARS 1
 LBT TEST FOR 8 BIT
 TRA NRK

*** DOES SZC 7414 CONDITION HOLD***

NRJ CAL CRYRG

LBT
 TRA NRK
 TRA NRH

NRK CAL MDR
 ARS 4

LBT TEST FOR 7 BIT**

TRA NRM

*** DOES SNA 7424 CONDITION HOLD***

NRL CAL ACCM
LGR 12

ZAC
LGL 12
TZE NRH

NRM CAL MDR
ARS 5

LBT TEST FOR 6 BIT
TRA NRD

*** DOES SPA 7444 CONDITION HOLD*****

CAL ACCM
LGR 11

LBT
TRA NRD
TRA NRH

NRD CALL PACK(PAC,1)

**** TEST FOR CLA 7401

NRH CAL MDR
LBT

TEST FOR 11 BIT HERE***

TRA NRP
STZ ACCM

** TEST FOR ORS 7402*****

NRP CAL MDR
ARS 1

LBT TEST FOR 10 BIT

TRA NRR
NRQ CAL ACCM

ORA SWRG
LGR 12

ZAC
LGL 12

*** TEST FOR STP 7500*****

NRR CAL MDR
ARS 6

LBT
TRA BEGIN

CLA BAFF+2

SUB =H MMRY

TNZ EXITT

BOBJJ CALL MMRY

EXITT TRA S.JXIT

***** END OF RS 2 INSTRUCTIONS*****

EJECT

* FOR STORAGE REFERENCE INSTRUCTIONS ENTER HERE

* TEST FOR SECTOR BIT

DCSR CLA IR
TZE ERROR

CLA MDR
LGR 6

LBT

TRA NOSBT

ZAC

PCS PAC,,4

SIM07100

SIM07110

SIM07120

SIM07130

SIM07140

SIM07150

SIM07160

SIM07170

SIM07180

SIM07190

SIM07200

SIM07210

SIM07220

SIM07230

SIM07240

SIM07250

SIM07260

SIM07270

SIM07280

SIM07290

SIM07300

SIM07310

SIM07320

SIM07330

SIM07340

SIM07350

SIM07360

SIM07370

SIM07380

SIM07390

SIM07400

SIM07410

SIM07420

SIM07430

SIM07440

SIM07450

SIM07460

SIM07470

SIM07480

SIM07490

SIM07500

SIM07510

SIM07520

SIM07530

SIM07540

SIM07550

SIM07560

SIM07570

SIM07580

SIM07590

SIM07600

SIM07610

SIM07620

SIM07630

SIM07640

	LGL	6		SIM07650
	TRA	ORINS		SIM07660
NOSBT	ZAC	'		SIM07670
	LGL	6		SIM07680
ORINS	ORA	BAAM		SIM07690
	SLW	MAR		SIM07700
* TEST FOR INDIRECT ADDRESSING				SIM07710
	CLA	MDR		SIM07720
	ARS	7		SIM07730
	LBT			SIM07740
	TRA	NINDR		SIM07750
* IF INDIRECTLY ADDRESSED MSG GOES INTO DEFER STATE				SIM07760
	CLA	=01		SIM07770
	STO	MSG		SIM07780
****	THE CLOCK STRIKES AGAIN*****			SIM07790
	CLA	CLOCK		SIM07800
	ADD	=1		SIM07810
	STO	CLOCK		SIM07820
	CALL	READ(MAR,MDR)		SIM07830
*****	TEST FOR AUTO INDEXING*****			SIM07840
	CLA	=020010		SIM07850
	CAS	MAR		SIM07860
	TRA	NOAUTO		SIM07870
	TRA	AUTO		SIM07880
	CLA	=020017 C(MAR).GE.20010		SIM07890
	CAS	MAR		SIM07900
	TRA	AUTO		SIM07910
	TRA	AUTO		SIM07920
	TRA	NOAUTO		SIM07930
AUTO	CLA	MDR		SIM07940
	ADD	=1		SIM07950
	STO	MDR		SIM07960
	CALL	WRITE(MAR,MDR)		SIM07970
NOAUTO	CLA	MDR		SIM07980
	ORA	BAAM		SIM07990
	STO	MAR		SIM08000
* MAR CONTAINS LOCATION OF OPERAND				SIM08010
* TO TEST IF IT IS JUMP INSTRUCTION				SIM08020
NINDR	CLA	IR		SIM08030
	SUB	=9		SIM08040
	TNZ	INDR		SIM08050
	CLA	MAR		SIM08060
	LGR	12		SIM08070
	ZAC			SIM08080
	LGL	12		SIM08090
	STO	PAC		SIM08100
	TRA	BEGIN		SIM08110
* SET MSG TO EXECUTE				SIM08120
INDR	CLA	=02		SIM08130
	STO	MSG		SIM08140
* BRING CONTENTS OF OPERAND TO MDR				SIM08150
	CALL	READ(MAR,MDR)		SIM08160
	LXA	IR,1		SIM08170
* CONTENTS OF IR ARE IN XR 1				SIM08180
	CALL	STRF		SIM08190

	TRA	BEGIN	SIM08200
*THIS BLOCK	CONTAINS	THE SUBPROGRAMS FOR THE	SIM08210
STRF	SAVE	1,2,4	SIM08220
*****	ALAS	THE CLOCK AGAIN****	SIM08230
	CLA	CLOCK	SIM08240
	ADD	=1	SIM08250
	STO	CLOCK	SIM08260
BABA	TRA*	BABA+12,1	SIM08270
	PZE	RATA	SIM08280
	PZE	RATB	SIM08290
	PZE	RATC	SIM08300
	PZE	RATD	SIM08310
	PZE	RATE	SIM08320
	PZE	RATF	SIM08330
	PZE	RATG	SIM08340
	PZE	RATH	SIM08350
	PZE	RATI	SIM08360
	PZE	RATJ	SIM08370
	PZE	RATK	SIM08380
	PZE	RATL	SIM08390
*	60	XCT EXECUTE	SIM08400
RATA	STZ	UTHANT+6	SIM08410
	TRA	BOAR	SIM08420
*	54	CAS COMPARE ACCUMULATOR AND SKIP	SIM08430
RATB	CALL	TCMPL(UTHANT+3,CASA)	SIM08440
	CALL	TCMPL(UTHANT+2,CASA+1)	SIM08450
	CLA	CASA	SIM08460
	CAS	CASA+1	SIM08470
	TRA	HAZRAT	SIM08480
	TRA	CASD	SIM08490
	TRA	CASE	SIM08500
CASD	CALL	PACK(UTHANT,1)	SIM08510
	TRA	HAZRAT	SIM08520
CASE	CALL	PACK(UTHANT,2)	SIM08530
	TRA	HAZRAT	SIM08540
*	50	JMS JUMP TO SUBROUTINE	SIM08550
RATC	CLA	UTHANT	SIM08560
	STO	UTHANT+2	SIM08570
	CALL	WRITE(UTHANT+1,UTHANT+2)	SIM08580
	CLA	UTHANT+1	SIM08590
	ADD	=1	SIM08600
	LGR	12	SIM08610
	ZAC		SIM08620
	LGL	12	SIM08630
	STO	UTHANT	SIM08640
RATD	TRA	HAZRAT	SIM08650
*	40	ISZ INCREMENT AND SKIP IF ZERO	SIM08660
RATE	CALL	PACK(UTHANT+2,1)	SIM08670
	CALL	WRITE(UTHANT+1,UTHANT+2)	SIM08680
	CAL	UTHANT+2	SIM08690
	TNZ	HAZRAT	SIM08700
	CALL	PACK(UTHANT,1)	SIM08710
	TRA	HAZRAT	SIM08720
*	34	RAD REPLACE ADD MEMORY	SIM08730
RATF	CAL	UTHANT+3	SIM08740

```

LGL      12                      SIM09300
STO      UTHANT+3                SIM09310
TRA      HAZRAT                  SIM09320
*        04      AND      LOGICAL      AND      SIM09330
RATL     CAL      UTHANT+3        SIM09340
        ANA      UTHANT+2        SIM09350
        SLW      UTHANT+3        SIM09360
HAZRAT   RETURN   STRF            SIM09370
        EJECT                     SIM09380
***** SIM09390
*** GIVE ERROR MESSAGE ON ILLEGAL INSTRUCTION AND QUIT AFTER SIM09400
*** GIVING A MEMORY DUMP***      SIM09410
ERROR    WRS      650,,3          SIM09420
        RCHA      *+1             SIM09430
        IORD      GALT1,,8        SIM09440
        TRA      BOBJJ            SIM09450
        EJECT                     SIM09460
***** SIM09470
** WHAT TO DO AT END OF EACH INSTRUCTION**** SIM09480
** AFTER END OF EXECUTION TEST FOR PROGRAM INTERRUPT** SIM09490
BEGIN    AXT      4,1             SIM09500
INTB     CLA      CLOCK            SIM09510
        SUB      KLOK+4,1         SIM09520
        CAS      =66667           SIM09530
        TRA      INTA             SIM09540
        TRA      INTA             SIM09550
        TRA      INTC             SIM09560
INTA     CLA      =1              SIM09570
        STO      DEVIK+4,1        SIM09580
INTC     TIX      INTB,1,1        SIM09590
*** WHEN UNIT IS READY CORRESPONDING DEVIK HAS 1 IN LAST BIT* SIM09600
        MIT      INTR             SIM09610
        TRA      BEGINN WHEN INTERRUPT NOT ON SIM09620
        AXT      4,1              SIM09630
INTE     ZAC      INTERRUPT ON    SIM09640
        PCS      DEVICE+4,1,4     SIM09650
        TNZ      INTD UNIT MASKED OUT SIM09660
        CLA      DEVICE+4,1       SIM09670
        LBT      SIM09680
        TRA      INTD UNIT IS NOT INTERRUPTING SIM09690
** SET MSG TO INTERRUPT MODE**  SIM09700
        CLA      =03              SIM09710
        STO      MSG              SIM09720
        CLA      PAC              SIM09730
        STO      8192             SIM09740
        CLA      =1              SIM09750
        STO      PAC              SIM09760
        ZAC      SIM09770
        SAC      DEVICE+4,1,5     SIM09780
        MSP      INTR TURN OF INTERRUPT SIM09790
        TRA      BEGINN          SIM09800
INTD     TIX      INTE,1,1        SIM09810
        TRA      BEGINN          SIM09820
        EJECT                     SIM09830
***** SIM09840

```



```

***** TO CONVERT FROM TWOS COMPLEMENTARY TO BINARY*****
TCMPL  SAVE    1,2,4
      CAL*    3,4
      LGR     11
      LBT
      TRA     *+2
      TRA     CASF
      LGL     11
      STO*    4,4
      TRA     CASG
CASF  LGL     11
      COM
      ADD     =1
      LGR     11
      ZAC
      LGL     11
      STO*    4,4
      MSM*    4,4
CASG  RETURN  TCMPL
      EJECT

*****
*** FOR INCREMENTING PAC WHEN DESIRED CALL THIS***
PACK  SAVE    1,2,4
      CAL*    3,4
      ADD     4,4
      LGR     12
      ZAC
      LGL     12
      STO*    3,4
      RETURN  PACK
      EJECT

*****
** TO WRITE ON TDC MEMORY***
WRITE SAVE    1,2,4
* TO  SAVE    ACM AND MQ
      STO     SAVED
      STQ     SAVEE    MQ SAVED
      CLA     3,4
      STA     MARA
      CLA*    4,4
MARA  STO*    **
      CLA     SAVED
      LDQ     SAVEE    MQ RESTORED
      RETURN  WRITE
      EJECT

*****
**** TO READ FROM TDC 12 MEMORY***
READ  SAVE    1,2,4
      STO     SAVEDX
      STQ     SAVEEX
      CLA     3,4
      STA     MARAX
MARAX CLA*    **
      STO*    4,4
      CLA     SAVEDX

      ACCUMULATOR SAVED
      MQ SAVED

      ACCUMULATOR RESTORED

```

SIM09850
 SIM09860
 SIM09870
 SIM09880
 SIM09890
 SIM09900
 SIM09910
 SIM09920
 SIM09930
 SIM09940
 SIM09950
 SIM09960
 SIM09970
 SIM09980
 SIM09990
 SIM10000
 SIM10010
 SIM10020
 SIM10030
 SIM10040
 SIM10050
 SIM10060
 SIM10070
 SIM10080
 SIM10090
 SIM10100
 SIM10110
 SIM10120
 SIM10130
 SIM10140
 SIM10150
 SIM10160
 SIM10170
 SIM10180
 SIM10190
 SIM10200
 SIM10210
 SIM10220
 SIM10230
 SIM10240
 SIM10250
 SIM10260
 SIM10270
 SIM10280
 SIM10290
 SIM10300
 SIM10310
 SIM10320
 SIM10330
 SIM10340
 SIM10350
 SIM10360
 SIM10370
 SIM10380
 SIM10390

LDQ SAVEEX MQ RESTORED
 RETURN READ
 EJECT

SIM10400
 SIM10410
 SIM10420

*****SIM10430

***** BEGIN DUMP OF TDC -12 MEMORY*****SIM10440

*****SIM10450

* NUMBER CONTAINS LOCATION COUNTERSIM10460

** OPWD CONTAINS CORRESPONDING OP CODESIM10470

MMRY SAVE 1,2,4SIM10480

CLA BLANKSIM10490

STO OPWDSIM10500

WRS 650,,3SIM10510

RCHA **1SIM10520

IORD LOCK,,3SIM10530

**** FOR ELECTRONIC REGISTERSSIM10540

*** FOR PAC MAR CRYRG SWRG***SIM10550

AXT 6,2SIM10560

RUBBER CLA UTHANT+6,2SIM10570

AXT 4,1SIM10580

LDQ UTHANT+6,2SIM10590

LGL 24SIM10600

ZAC SIM10610

PENCIL ALS 3SIM10620

LGL 3SIM10630

TIX PENCIL,1,1SIM10640

ALS 6SIM10650

PCS BLANK,,5SIM10660

ALS 6SIM10670

PCS BLANK,,5SIM10680

STO NUMBR+6,2SIM10690

TIX RUBBER,2,1SIM10700

***** FOR ACCM AND MDR*****SIM10710

AXT 2,2SIM10720

PNCH CAL UTHANT+4,2SIM10730

STO GLIDERSIM10740

CALL TCMLP(GLIDER, GLIDER)SIM10750

AXT 4,1SIM10760

LDQ GLIDERSIM10770

LGL 24SIM10780

ZAC SIM10790

LAMP ALS 3SIM10800

LGL 3SIM10810

TIX LAMP,1,1SIM10820

ALS 6SIM10830

PCS BLANK,,5SIM10840

STO NUMBR+4,2SIM10850

CAL GLIDERSIM10860

PBT SIM10870

TRA ONESIM10880

CLA MINUSSIM10890

SAC NUMBR+4,2,0SIM10900

TRA KEYSIM10910

ONE CLA BLANKSIM10920

SAC NUMBR+4,2,0SIM10930

KEY TIX PNCH,2,1SIM10940

*****	FOR	CLOCK*****		SIM10950
	AXT	2,2		SIM10960
	LDQ	CLOCK		SIM10970
KEYX	AXT	6,1		SIM10980
	ZAC			SIM10990
SCREW	ALS	3		SIM11000
	LGL	3		SIM11010
	TIX	SCREW,1,1		SIM11020
	STO	NUMBR+9,2		SIM11030
	TIX	KEYX,2,1		SIM11040
	CLA	BLANK		SIM11050
	STO	NUMBR+6		SIM11060
	WRS	650,,3		SIM11070
	RCHA	*+1		SIM11080
	IORD	TITLE,,9		SIM11090
	WRS	650,,3		SIM11100
	RCHA	*+1		SIM11110
	IORD	NUMBR,,9		SIM11120
***	FOR CORE	MEMORY 0000 THRU 7777 OCTAL		SIM11130
	AXT	4096,4		SIM11140
CAMEL	AXT	16,2		SIM11150
	TXI	*+1,4,1		SIM11160
	CLA	ZERO		SIM11170
	ADD	=020		SIM11180
	STO	ZERO		SIM11190
	LGR	12		SIM11200
	AXT	4,1		SIM11210
	ZAC			SIM11220
ANT	ALS	3		SIM11230
	LGL	3		SIM11240
	TIX	ANT,1,1		SIM11250
	ALS	6		SIM11260
	PCS	BLANK,,5		SIM11270
	ALS	6		SIM11280
	PCS	BLANK,,5		SIM11290
	SLW	NUMBR		SIM11300
BACK	TXI	*+1,4,-1		SIM11310
	AXT	42,1		SIM11320
	CAL	12288,4		SIM11330
COMPR	CAS	IOCS+42,1		SIM11340
	TIX	COMPR,1,1		SIM11350
	TRA	*+2		SIM11360
	TIX	COMPR,1,1		SIM11370
	PXA	,1		SIM11380
	SUB	=1		SIM11390
	TNZ	REGSTR GO FOR REGISTER SET		SIM11400
	CLA	=07500		SIM11410
	SUB	12288,4		SIM11420
	TNZ	MEMREF		SIM11430
***	HANDLE	NON STORAGE REFERENCE HERE****		SIM11440
REGSTR	CAL	REGSET+42,1		SIM11450
	SLW	OPWD+17,2		SIM11460
	TRA	BABAR		SIM11470
***	HANDLE	STORAGE REFERENCE HERE		SIM11480
MEMREF	CLA	12288,4		SIM11490

LGR	8		SIM11500
TZE	ORDNY	GO FOR ORDINARY NUMBERS	SIM11510
CAS	TRTEEN		SIM11520
TRA	ORDNY		SIM11530
TRA	ORDNY		SIM11540
PAX	,1		SIM11550
CLA	STGREF+12,1		SIM11560
STO	OPWD+17,2		SIM11570
****	TEST	FOR SECTOR BIT	SIM11580
CLA	12288,4		SIM11590
LGR	6		SIM11600
LBT			SIM11610
TRA	NOSKTR		SIM11620
CLA	RECTOR		SIM11630
SAC	OPWD+17,2,4		SIM11640
TRA	KOBLAI		SIM11650
NOSKTR	CLA	BLANK	SIM11660
SAC	OPWD+17,2,4		SIM11670
***	TEST	FOR INDIRECT ADDRESSING	SIM11680
KOBLAI	CLA	12288,4	SIM11690
ARS	7		SIM11700
LBT			SIM11710
TRA	NOIMDR		SIM11720
CLA	STAR		SIM11730
SAC	OPWD+17,2,3		SIM11740
TRA	KHAN		SIM11750
NOIMDR	CLA	BLANK	SIM11760
SAC	OPWD+17,2,3		SIM11770
KHAN	CLA	BLANK	SIM11780
SAC	OPWD+17,2,5		SIM11790
****	THE	CORRECT OP CODE HAS BEEN SELECTED AND PUT	SIM11800
***	TO	GET THREE LEADING ZEROES	SIM11810
BABAR	AXT	4,1	SIM11820
LDQ	12288,4		SIM11830
LGL	24		SIM11840
ZAC			SIM11850
SHIFT	ALS	3	SIM11860
LGL	3		SIM11870
TIX	SHIFT,1,1		SIM11880
ALS	6		SIM11890
PCS	BLANK,,5		SIM11900
ALS	6		SIM11910
PCS	BLANK,,5		SIM11920
SLW	NUMBR+17,2		SIM11930
TRA	TATA		SIM11940
**	ENTER	HERE FOR SIMPLE NUMBERS	SIM11950
ORDNY	CLA	BLANK	SIM11960
STO	OPWD+17,2		SIM11970
CLA	12288,4		SIM11980
STO	GLIDER		SIM11990
CALL	TCMPL(GLIDER, GLIDER)		SIM12000
AXT	4,1		SIM12010
LDQ	GLIDER		SIM12020
LGL	24		SIM12030
ZAC			SIM12040

VARIAC	ALS	3		SIM12050
	LGL	3		SIM12060
	TIX	VARIAC,1,1		SIM12070
	ALS	6		SIM12080
	PCS	BLANK,,5		SIM12090
	STO	NUMBR+17,2		SIM12100
	CAL	GLIDER		SIM12110
	PBT			SIM12120
	TRA	PLS		SIM12130
	CLA	MINUS		SIM12140
	SAC	NUMBR+17,2,0		SIM12150
	TRA	TATA		SIM12160
PLS	CLA	BLANK		SIM12170
	SAC	NUMBR+17,2,0		SIM12180
*****	TEST	FOR LOOPING		SIM12190
TATA	TIX	BACK,2,1		SIM12200
	CALL	DAMP		SIM12210
	TIX	CAMEL,4,1		SIM12220
	RETURN	MMRY		SIM12230
***	GIVE	SUITABLE MESSAGE IF MANY LICATIONS HAVE SAME		SIM12240
*****	CONTENTS	STORED IN THEM*****		SIM12250
***	DAMP	SUBROUTINE TAKES CARE OF THAT***		SIM12260
DAMP	SAVE	1,2,4		SIM12270
	AXT	16,1		SIM12280
BOOK	CLA	NUMBR+1		SIM12290
	SUB	NUMBR+17,1		SIM12300
	TNZ	DAMPB CONTENTS OF NUMBER+1 THRU +16 ARE NOT SAME		SIM12310
	TIX	BOOK,1,1		SIM12320
*****	IN	CASE CONTENTS ARE SAME		SIM12330
	PLT	TEA		SIM12340
	TRA	DAMPD		SIM12350
***	WHEN	TEA IS PLUS		SIM12360
	CLA	NUMBR		SIM12370
	STO	SPARE		SIM12380
	CLA	ZERO		SIM12390
	ADD	=017		SIM12400
	STO	SPARE+4		SIM12410
	CLA	NUMBR+1		SIM12420
	STO	SPARE+2		SIM12430
	CLA	OPWD+1		SIM12440
	STO	SPARE+3		SIM12450
	MSM	TEA		SIM12460
	CLA	SPARE+4		SIM12470
	SUB	=07777		SIM12480
	TZE	DAMPF		SIM12490
	TRA	CLIVE		SIM12500
****	WHEN	TEA IS MINUS		SIM12510
DAMPD	CLA	NUMBR+1		SIM12520
	SUB	SPARE+2		SIM12530
	TNZ	DAMPF		SIM12540
****	WHEN	NUMBR+1 AND SPARE+2 ARE SAME***		SIM12550
	CLA	SPARE+4		SIM12560
	ADD	=020		SIM12570
	STO	SPARE+4		SIM12580
	CLA	SPARE+4		SIM12590

***	*****	DAMPF	BOOKA	***	*****	DAMPB	BOOKC	
		SUB						SIM12600
		TZE						SIM12610
		TRA						SIM12620
								SIM12630
		WHEN						SIM12640
		SPARE+2						SIM12650
		AXT						SIM12660
		LDQ						SIM12670
		LGL						SIM12680
		ZAC						SIM12690
		ALS						SIM12700
		LGL						SIM12710
		TIX						SIM12720
		ALS						SIM12730
		PCS						SIM12740
		ALS						SIM12750
		PCS						SIM12760
		SLW						SIM12770
		CLA						SIM12780
		STO						SIM12790
		CLA						SIM12800
		STO						SIM12810
		CLA						SIM12820
		STO						SIM12830
		CLA						SIM12840
		STO						SIM12850
		WRS						SIM12860
		RCHA						SIM12870
		IORD						SIM12880
		CLA						SIM12890
		SUB						SIM12900
		TZE						SIM12910
								SIM12920
		TO						SIM12930
		SAVE						SIM12940
								SIM12950
		CLA						SIM12960
		STO						SIM12970
		CLA						SIM12980
		ADD						SIM12990
		STO						SIM13000
		CLA						SIM13010
		STO						SIM13020
		CLA						SIM13030
		STO						SIM13040
		CLA						SIM13050
		SUB						SIM13060
		TZE						SIM13070
		TRA						SIM13080
								SIM13090
		WHEN						SIM13100
		MIT						SIM13110
		TRA						SIM13120
		AXT						SIM13130
		LDQ						SIM13140
		LGL						
		ZAC						
		ALS						
		LGL						
		TIX						

ALS	6		SIM13150
PCS	BLANK,,5		SIM13160
ALS	6		SIM13170
PCS	BLANK,,5		SIM13180
SLW	SPARE+1		SIM13190
CLA	SPARE		SIM13200
STO	MSSGE+2		SIM13210
CLA	SPARE+1		SIM13220
STO	MSSGE+4		SIM13230
CLA	SPARE+2		SIM13240
STO	MSSGE+7		SIM13250
CLA	SPARE+3		SIM13260
STO	MSSGE+9		SIM13270
WRS	650,,3		SIM13280
RCHA	*+1		SIM13290
IORD	MSSGE,,10		SIM13300
***** MAKE	TEA PLUS*****		SIM13310
MSP	TEA		SIM13320
***** WRITE	OUT *****		SIM13330
DAMPG	WRS	650,,3	SIM13340
	RCHA	*+1	SIM13350
	IORD	NUMBR,,17	SIM13360
	WRS	650,,3	SIM13370
	RCHA	*+1	SIM13380
	IORD	OPWD,,17	SIM13390
CLIVE	RETURN	DAMP	SIM13400
	EJECT.		SIM13410
*****			SIM13420
CHAR	PZE	TSFQA	SIM13430
	PZE	TSFQ	SIM13440
	PZE	KSFQA	SIM13450
	PZE	KSFQ	SIM13460
	PZE	IOFQ	SIM13470
DO	PZE	TCFQA	SIM13480
	PZE	TCFQ	SIM13490
	PZE	KCCQA	SIM13500
	PZE	KCCQ	SIM13510
	PZE	IONQ	SIM13520
EK	PZE	TPCQA	SIM13530
	PZE	TPCQ	SIM13540
	PZE	KRSQA	SIM13550
	PZE	KRSQ	SIM13560
	PZE	SMKQ	SIM13570
BUFF01	OCT	726060606060	SIM13580
	BCI	8,	SIM13590
	BCI	4,	SIM13600
BUFF02	OCT	726060606060.	SIM13610
	BCI	8,	SIM13620
	BCI	4,	SIM13630
BUFF03	OCT	726060606060	SIM13640
	BCI	8,	SIM13650
	BCI	6,	SIM13660
BUFF04	OCT	726060606060	SIM13670
	BCI	8,	SIM13680
	BCI	6,	SIM13690

UNIT 03

UNIT 04

INTR	OCT	0				SIM13700
KLOK	OCT	0,0,0,0				SIM13710
DEVIK	OCT	1,1,1,1				SIM13720
UNIT	BSS	1				SIM13730
FLIP	BSS	1				SIM13740
RCMRK	OCT	72				SIM13750
LINFID	BCI	1,				SIM13760
CARRET	BCI	1,				SIM13770
TELTYP	OCT	301,302,303,304,305,306,307,310,311,312,				SIM13780
	ETC	313,314,315,316,317,320,321,322,323,324,				SIM13790
	ETC	325,326,327,330,331,332,260,261,262,263,				SIM13800
	ETC	264,265,266,267,270,271,244,247,250,				SIM13810
	ETC	251,252,253,254,255,256,257,275,000,				SIM13820
	ETC	212,215				SIM13830
IBM	OCT	21,22,23,24,25,26,27,30,31,41,				SIM13840
	ETC	42,43,44,45,46,47,50,51,62,63,				SIM13850
	ETC	64,65,66,67,70,71,00,01,02,03,				SIM13860
	ETC	04,05,06,07,10,11,53,14,74,				SIM13870
	ETC	34,54,20,73,40,33,61,13,60,				SIM13880
	ETC	76,77				SIM13890
SETB	BCI	6,INPUT WANTED ON UNIT		.PAUSE		SIM13900
HARSHA	BSS	1				SIM13910
BAFF	BSS	3				SIM13920
MSSGE	BCI	9,LOCATION	THRU	ALL CONTAIN		SIM13930
SPARE	BCI	5,				SIM13940
TEA	BSS	1				SIM13950
CLOCK	OCT	0				SIM13960
LOCK	BCI	3,DUMP OF TDC MEMORY				SIM13970
TITLE	BCI	9,PAC MAR MDR ACCM CRYRG SWRG		CLOCK		SIM13980
BLANK	BCI	1,				SIM13990
REGSET	BCI	1,SMK	6401			SIM14000
	BCI	1,ION	6402			SIM14010
	BCI	1,IOF	6404			SIM14020
	BCI	1,KRS	6411			SIM14030
	BCI	1,KCC	6412			SIM14040
	BCI	1,KRB	6413			SIM14050
	BCI	1,KSF	6414			SIM14060
	BCI	1,KRC	6421			SIM14070
	BCI	1,KCS	6422			SIM14080
	BCI	1,KRP	6423			SIM14090
	BCI	1,KSP	6424			SIM14100
	BCI	1,TPC	6431			SIM14110
	BCI	1,TCF	6432			SIM14120
	BCI	1,TLS	6433			SIM14130
	BCI	1,TSF	6434			SIM14140
	BCI	1,TPS	6441			SIM14150
	BCI	1,TCP	6442			SIM14160
	BCI	1,TLC	6443			SIM14170
	BCI	1,TSP	6444			SIM14180
	BCI	1,NOP	7000			SIM14190
	BCI	1,CLA	7001			SIM14200
	BCI	1,CLC	7002			SIM14210
	BCI	1,CAR	7010			SIM14220
	BCI	1,CAL	7020			SIM14230
	BCI	1,CTR	7014			SIM14240

	BCI	1,CTL	7024		SIM14250
	BCI	1,CMC	7040		SIM14260
	BCI	1,CMA	7100		SIM14270
	BCI	1,IAC	7200		SIM14280
	BCI	1,CIA	7300		SIM14290
	BCI	1,STC	7042		SIM14300
	BCI	1,STA	7101		SIM14310
	BCI	1,CLA	7401		SIM14320
	BCI	1,ORS	7402		SIM14330
	BCI	1,SKP	7404		SIM14340
	BCI	1,SNC	7410		SIM14350
	BCI	1,SZC	7414		SIM14360
	BCI	1,SZA	7420		SIM14370
	BCI	1,SNA	7424		SIM14380
	BCI	1,SMA	7440		SIM14390
	BCI	1,SPA	7444		SIM14400
	BCI	1,STP	7500		SIM14410
STGREF	BCI	1,XCT	60		SIM14420
	BCI	1,CAS	54		SIM14430
	BCI	1,JMS	50		SIM14440
	BCI	1,JMP	44		SIM14450
	BCI	1,ISZ	40		SIM14460
	BCI	1,RAD	34		SIM14470
	BCI	1,SUB	30		SIM14480
	BCI	1,ADD	24		SIM14490
	BCI	1,SAC	20		SIM14500
	BCI	1,LAC	14		SIM14510
	BCI	1,XOR	10		SIM14520
	BCI	1,AND	04		SIM14530
OPWD	BSS	17			SIM14540
NUMBR	BSS	17			SIM14550
ZERO	OCT	-20			SIM14560
GLIDER	BSS	1			SIM14570
STAR	BCI	1,*****			SIM14580
RECTOR	BCI	1,111111			SIM14590
MINUS	BCI	1,-----			SIM14600
TRTEEN	DEC	13			SIM14610
SAVEEX	BSS	1			SIM14620
SAVEDX	BSS	1			SIM14630
SAVEE	BSS	1			SIM14640
SAVED	BSS	1			SIM14650
GALTI	BCI	8, ILLEGAL TDC INSTRUCTION.		JOB TERMINATED	SIM14660
CASA	BSS	2			SIM14670
ETONE	BSS	81			SIM14680
GUMP	BCI	7, LOCATION COUNTER JUMP		THRU	SIM14690
VAS	BSS	2			SIM14700
BUFF	BSS	27			SIM14710
DRAKE	PZE	ETONE+3			SIM14720
BING	BCI	5, ILLEGAL INPUT. UNABLE TO READ.			SIM14730
CUFF	BSS	2			SIM14740
IR	BSS	1			SIM14750
SENSE	BSS	1			SIM14760
RDREDR	BCI	3, 21556 CD RD ERR			SIM14770
IOCS	OCT	6401, 6402, 6404, 6411, 6412, 6413, 6414,			SIM14780
ETC		6421, 6422, 6423, 6424, 6431, 6432, 6433, 6434,			SIM14790